RM0521
Reference manual

STM32WBA2 advanced Arm®-based 32-bit MCUs

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32WBA2 microprocessor memory and peripherals.

The STM32WBA2 is a family of microprocessors with different packages and peripherals.

Refer to the corresponding datasheets for ordering information, mechanical and electrical device characteristics.

For information on the Arm®, Cortex®-M33 core, refer to the corresponding Technical Reference Manuals available on http://infocenter.arm.com.

The STM32WBA2 microprocessors include ST state-of-the-art patented technology.

Related documents

- STM32WBA2 datasheet (DS15003)
- STM32WBA2 errata sheet (ES0671)

Contents

1 Documentation conventions 66
1.1 General information 66
1.2 List of abbreviations for registers 66
1.3 Register reset value 67
1.4 Glossary 67
1.5 Availability of peripherals 67

2 Memory and bus architecture 68
2.1 System architecture 68
2.1.1 CPU C-bus 69
2.1.2 CPU S-bus 69
2.1.3 LPDMA1-bus 69
2.1.4 Bus matrix 70
2.1.5 AHB/APB bridges 70
2.2 TrustZone® security architecture 70
2.2.1 Default TrustZone® security state 72
2.2.2 TrustZone® peripheral classification 72
2.3 Memory organization 75
2.3.1 Introduction 75
2.3.2 Memory map and register boundary addresses 76
2.3.3 Embedded SRAM 81
2.3.4 Flash memory overview 82

3 Boot modes 83

4 System security 85
4.1 Key security features 85
4.2 Secure boot 86
4.2.1 Unique boot entry and BOOT_LOCK 86
4.2.2 Immutable root of trust in system flash memory 86
4.3 Secure update 86
4.4 Resource isolation using TrustZone 87
4.4.1 TrustZone security architecture 88

4.4.2 Armv8-M security extension of Cortex-M33 88
4.4.3 Memory and peripheral allocation using IDAU/SAU 88
4.4.4 Memory and peripheral allocation using GTZC 90
4.4.5 Managing security in TrustZone-aware peripherals 94
4.4.6 Activating TrustZone security 100
4.4.7 Deactivating TrustZone security 101
4.5 Other resource isolations 101
4.5.1 Temporal isolation using secure hide protection (HDP) 101
4.5.2 RSSLIB functions 102
4.5.3 Resource isolation using Cortex privileged mode 103
4.6 Secure execution 107
4.6.1 Memory protection unit (MPU) 107
4.6.2 Embedded flash memory write protection 108
4.6.3 Tamper detection and response 108
4.7 Secure storage 110
4.7.1 Unique ID 110
4.8 Crypto engines 110
4.8.1 Crypto engines features 111
4.8.2 On-the-fly decryption engine (OTFDEC) 111
4.9 Product life cycle 112
4.9.1 Life cycle management with readout protection (RDP) 113
4.9.2 Recommended option byte settings 116
4.10 Access controlled debug 116
4.10.1 Debug protection with readout protection (RDP) 116
4.11 Software intellectual property protection and collaborative development 117
4.11.1 Software intellectual property protection with RDP 118
4.11.2 Software intellectual property protection with OTFDEC 118
4.11.3 Other software intellectual property protections 119

5 Global TrustZone® controller (GTZC) 120
5.1 Introduction 120
5.2 GTZC main features 120
5.3 GTZC implementation 122
5.4 GTZC functional description 123
5.4.1 GTZC block diagram 123
5.4.2 Illegal access definition 124

5.4.3 TrustZone security controller (TZSC) . . . 125
5.4.4 Memory protection controller - block based (MPCBB) . . . 126
5.4.5 TrustZone illegal access controller (TZIC) . . . 127
5.4.6 Power-on/reset state . . . 127
5.5 GTZC interrupts . . . 128
5.6 GTZC1 TZSC registers . . . 128
5.6.1 GTZC1 TZSC control register (GTZC1_TZSC_CR) . . . 128
5.6.2 GTZC1 TZSC secure configuration register 1 (GTZC1_TZSC_SECCFGR1) . . . 129
5.6.3 GTZC1 TZSC secure configuration register 2 (GTZC1_TZSC_SECCFGR2) . . . 130
5.6.4 GTZC1 TZSC secure configuration register 3 (GTZC1_TZSC_SECCFGR3) . . . 131
5.6.5 GTZC1 TZSC privilege configuration register 1 (GTZC1_TZSC_PRIVCFGR1) . . . 133
5.6.6 GTZC1 TZSC privilege configuration register 2 (GTZC1_TZSC_PRIVCFGR2) . . . 134
5.6.7 GTZC1 TZSC privilege configuration register 3 (GTZC1_TZSC_PRIVCFGR3) . . . 135
5.6.8 GTZC1 TZSC memory 1 subregion z watermark configuration register (GTZC1_TZSC_MPCWM1zCFGR) (z = A to B) . . . 136
5.6.9 GTZC1 TZSC memory 1 subregion A watermark register (GTZC1_TZSC_MPCWM1AR) . . . 137
5.6.10 GTZC1 TZSC memory 1 subregion B watermark register (GTZC1_TZSC_MPCWM1BR) . . . 139
5.6.11 GTZC1 TZSC register map . . . 140
5.7 GTZC1 TZIC registers . . . 142
5.7.1 GTZC1 TZIC interrupt enable register 1 (GTZC1_TZIC_IER1) . . . 142
5.7.2 GTZC1 TZIC interrupt enable register 2 (GTZC1_TZIC_IER2) . . . 143
5.7.3 GTZC1 TZIC interrupt enable register 3 (GTZC1_TZIC_IER3) . . . 144
5.7.4 GTZC1 TZIC interrupt enable register 4 (GTZC1_TZIC_IER4) . . . 145
5.7.5 GTZC1 TZIC status register 1 (GTZC1_TZIC_SR1) . . . 147
5.7.6 GTZC1 TZIC status register 2 (GTZC1_TZIC_SR2) . . . 148
5.7.7 GTZC1 TZIC status register 3 (GTZC1_TZIC_SR3) . . . 149
5.7.8 GTZC1 TZIC status register 4 (GTZC1_TZIC_SR4) . . . 150
5.7.9 GTZC1 TZIC flag clear register 1 (GTZC1_TZIC_FCR1) . . . 152
5.7.10 GTZC1 TZIC flag clear register 2 (GTZC1_TZIC_FCR2) . . . 152
5.7.11 GTZC1 TZIC flag clear register 3 (GTZC1_TZIC_FCR3) . . . 154
5.7.12 GTZC1 TZIC flag clear register 4 (GTZC1_TZIC_FCR4) . . . 155

5.7.13 GTZC1 TZIC register map 157
5.8 GTZC1 MPCBB registers 158
5.8.1 GTZC1 MPCBB control register (GTZC1_MPCBBx_CR) (x = 1 to 2) . 158
5.8.2 GTZC1 MPCBB1 configuration lock register (GTZC1_MPCBB1_CFGLOCK) 159
5.8.3 GTZC1 MPCBB2 configuration lock register (GTZC1_MPCBB2_CFGLOCK) 159
5.8.4 GTZC1 MPCBB security configuration for superblock n register (GTZC1_MPCBBx_SECCFGRn) (x = 1 to 2) 160
5.8.5 GTZC1 MPCBB1 security configuration for superblock n register (GTZC1_MPCBB1_SECCFGRn) 160
5.8.6 GTZC1 MPCBB privileged configuration for superblock n register (GTZC1_MPCBBx_PRIVCFGRn) (x = 1 to 2) 161
5.8.7 GTZC1 MPCBB1 privileged configuration for superblock n register (GTZC1_MPCBB1_PRIVCFGRn) 161
5.8.8 GTZC1 MPCBB1 register map 162

6 RAMs configuration controller (RAMCFG) 164
6.1 Introduction 164
6.2 RAMCFG main features 164
6.3 RAMCFG functional description 164
6.3.1 Internal SRAM features 164
6.3.2 Internal SRAM parity 165
6.3.3 Internal SRAM write protection 166
6.3.4 Internal SRAM read access latency 166
6.3.5 Internal SRAM erase 167
6.4 RAMCFG low-power modes 167
6.5 RAMCFG interrupts 167
6.6 RAMCFG registers 168
6.6.1 RAMCFG SRAM1 control register (RAMCFG_M1CR) 168
6.6.2 RAMCFG SRAM1 interrupt status register (RAMCFG_M1ISR) 169
6.6.3 RAMCFG SRAM erase key register (RAMCFG_MxERKEYR) 169
6.6.4 RAMCFG SRAM2 control register (RAMCFG_M2CR) 170
6.6.5 RAMCFG SRAM2 interrupt enable register (RAMCFG_M2IER) 171
6.6.6 RAMCFG SRAM2 interrupt status register (RAMCFG_M2ISR) 171
6.6.7 RAMCFG SRAM2 parity error address register (RAMCFG_M2PEAR) 172
6.6.8 RAMCFG SRAM2 interrupt clear register (RAMCFG_M2ICR) 173
6.6.9 RAMCFG SRAM2 write protection register 1 . .(RAMCFG_M2WPR1) 173

6.6.10 RAMCFG register map 174

7 Embedded flash memory (FLASH) 175

7.1 Introduction 175

7.2 FLASH main features 175

7.3 FLASH functional description 175

7.3.1 Flash memory organization 175

7.3.2 Error code correction (ECC) 177

7.3.3 Read access latency 177

7.3.4 Flash power-down mode 179

7.3.5 Flash memory program and erase operations 180

7.3.6 Flash memory erase sequences 182

7.3.7 Flash memory programming sequences 183

7.3.8 Flash memory programming erases suspend 186

7.3.9 Flash memory operation abort 187

7.3.10 Flash memory endurance 187

7.3.11 Flash memory errors flags 188

7.3.12 Power-down during programming or erase operations 189

7.3.13 Interruption during programming or erase operations 190

7.4 FLASH option bytes 191

7.4.1 Option bytes description 191

7.4.2 Option bytes programming 191

7.5 FLASH TrustZone security and privilege protections 194

7.5.1 Trustzone security protection 194

7.5.2 Watermark-based secure flash memory area protection 195

7.5.3 Secure hide protection (HDP) 195

7.5.4 Secure hide protection extension (HDP extension) 196

7.5.5 Block-based secure flash memory area protection 198

7.5.6 Flash security attribute state 199

7.5.7 Block-based privileged flash memory area protection 199

7.5.8 Flash memory registers privileged and unprivileged modes 200

7.6 Flash memory protection 200

7.6.1 Write protection (WRP) 200

7.6.2 Readout protection (RDP) 202

7.7 Summary of flash memory and registers access control 211

7.8 FLASH interrupts 215

7.9 FLASH registers 217
7.9.1 FLASH access control register (FLASH_ACR) 217
7.9.2 FLASH key register (FLASH_NSKEYR) 218
7.9.3 FLASH secure key register (FLASH_SECKEYR) 219
7.9.4 FLASH option key register (FLASH_OPTKEYR) 219
7.9.5 FLASH power-down key register (FLASH_PDKEYR) 220
7.9.6 FLASH status register (FLASH_NSSR) 220
7.9.7 FLASH secure status register (FLASH_SECSR) 222
7.9.8 FLASH control register (FLASH_NSCR1) 224
7.9.9 FLASH secure control register (FLASH_SECCR1) 226
7.9.10 FLASH ECC register (FLASH_ECCR) 227
7.9.11 FLASH operation status register (FLASH_OPSR) 228
7.9.12 FLASH control 2 register (FLASH_NSCR2) 229
7.9.13 FLASH secure control 2 register (FLASH_SECCR2) 230
7.9.14 FLASH option register (FLASH_OPTR) 230
7.9.15 FLASH boot address 0 register (FLASH_NSBOOTADD0R) 232
7.9.16 FLASH boot address 1 register (FLASH_NSBOOTADD1R) 233
7.9.17 FLASH secure boot address 0 register (FLASH_SECBOOTADD0R) . 233
7.9.18 FLASH secure watermark register 1 (FLASH_SECWMR1) 234
7.9.19 FLASH secure watermark register 2 (FLASH_SECWMR2) 235
7.9.20 FLASH WRP area A address register (FLASH_WRPAR) 235
7.9.21 FLASH WRP area B address register (FLASH_WRPBR) 236
7.9.22 FLASH secure block based register x (FLASH_SECBBRx) 237
7.9.23 FLASH secure HDP control register (FLASH_SECHDPCR) 237
7.9.24 FLASH privilege configuration register (FLASH_PRIFCFGR) 238
7.9.25 FLASH secure HDP extension register (FLASH_SECHDPEXTR) . . . 239
7.9.26 FLASH privilege block based register x (FLASH_PRIVBBRx) 239
7.9.27 FLASH OEM1 key register 1 (FLASH_OEM1KEYR1) 240
7.9.28 FLASH OEM1 key register 2 (FLASH_OEM1KEYR2) 240
7.9.29 FLASH OEM1 key register 3 (FLASH_OEM1KEYR3) 241
7.9.30 FLASH OEM1 key register 4 (FLASH_OEM1KEYR4) 241
7.9.31 FLASH OEM2 key register 1 (FLASH_OEM2KEYR1) 242
7.9.32 FLASH OEM2 key register 2 (FLASH_OEM2KEYR2) 242
7.9.33 FLASH OEM2 key register 3 (FLASH_OEM2KEYR3) 243
7.9.34 FLASH OEM2 key register 4 (FLASH_OEM2KEYR4) 243
7.9.35 FLASH OEM key status register (FLASH_OEMKEYSR) 244
7.9.36 FLASH register map 245

8 Instruction cache (ICACHE) 249
8.1 ICACHE introduction 249
8.2 ICACHE main features 249
8.3 ICACHE implementation 250
8.4 ICACHE functional description 250
8.4.1 ICACHE block diagram 251
8.4.2 ICACHE reset and clocks 251
8.4.3 ICACHE TAG memory 252
8.4.4 Direct-mapped ICACHE (1-way cache) 253
8.4.5 ICACHE enable 254
8.4.6 Cacheable and noncacheable traffic 254
8.4.7 Address remapping 255
8.4.8 Cacheable accesses 257
8.4.9 Dual-master cache 258
8.4.10 ICACHE security 258
8.4.11 ICACHE maintenance 258
8.4.12 ICACHE performance monitoring 259
8.4.13 ICACHE boot 259
8.5 ICACHE low-power modes 259
8.6 ICACHE error management and interrupts 260
8.7 ICACHE registers 260
8.7.1 ICACHE control register (ICACHE_CR) 260
8.7.2 ICACHE status register (ICACHE_SR) 261
8.7.3 ICACHE interrupt enable register (ICACHE_IER) 262
8.7.4 ICACHE flag clear register (ICACHE_FCR) 262
8.7.5 ICACHE hit monitor register (ICACHE_HMONR) 263
8.7.6 ICACHE miss monitor register (ICACHE_MMONR) 263
8.7.7 ICACHE region x configuration register (ICACHE_CRRx) 263
8.7.8 ICACHE register map 264

9 Radio system 266
9.1 Introduction 266
9.2 Main features 266
9.3 2.4 GHz RADIO implementation 267
9.4 Functional description 267
9.4.1 Block diagram 267

9.4.2 Pins and internal signals 267
9.4.3 Transmit output power 268
9.4.4 Bluetooth AoA/AoD and channel sounding antenna selection 269
9.4.5 RXTX data SRAM access 269
9.5 Low-power modes 270

10 PTA converter (PTACONV) 271
10.1 PTACONV introduction 271
10.2 PTACONV main features 271
10.3 PTACONV functional description 271
10.3.1 PTACONV block diagram 272
10.3.2 PTACONV pins and internal signals 272
10.3.3 PTACONV protocols 273
10.3.4 PTACONV interface with the 2.4 GHz RADIO 276
10.4 PTACONV low-power modes 276
10.5 PTACONV registers 278
10.5.1 PTACONV active control register (PTACONV_ACTCR) 278
10.5.2 PTACONV priority control register (PTACONV_PRICR) 279
10.5.3 PTACONV control register (PTACONV_CR) 279
10.5.4 PTACONV register map 281

11 Power control (PWR) 282
11.1 Introduction 282
11.2 PWR main features 282
11.3 PWR pins and internal signals 283
11.4 PWR power supplies and supply domains 285
11.4.1 External power supplies 286
11.4.2 Application power supply schemes 287
11.4.3 Power-up and power-down power sequences 288
11.4.4 Independent analog peripherals supply 288
11.4.5 Independent USB transceiver supply 288
11.4.6 Independent RADIO peripheral supply 288
11.4.7 Backup domain 288
11.4.8 Internal regulators 289
11.5 PWR system supply voltage regulation 289
11.5.1 SMPS and LDO embedded regulators 289

11.5.2 LDO and SMPS versus reset, voltage scaling, and low-power modes 289
11.5.3 LDO and SMPS fast startup 290
11.5.4 Dynamic voltage scaling management 290
11.5.5 2.4 GHz RADIO PA regulator 291
11.6 PWR power supply supervision 292
11.6.1 Brownout reset (BOR) 292
11.6.2 Programmable voltage detector (PVD) 293
11.7 PWR power management 293
11.7.1 PWR power modes 293
11.7.2 PWR background autonomous mode (BAM) 302
11.7.3 PWR Run mode 304
11.7.4 PWR low-power modes 305
11.7.5 PWR Sleep mode 306
11.7.6 PWR Stop 0 mode 307
11.7.7 PWR Stop 1 mode 311
11.7.8 PWR Stop 2 mode 312
11.7.9 PWR Stop 3 mode 314
11.7.10 PWR Standby mode 316
11.7.11 Power modes output pins 319
11.8 PWR security and privileged protection 320
11.8.1 PWR security protection 320
11.8.2 PWR privileged protection 321
11.9 PWR interrupts 322
11.10 PWR registers 323
11.10.1 PWR control register 1 (PWR_CR1) 323
11.10.2 PWR control register 2 (PWR_CR2) 324
11.10.3 PWR control register 3 (PWR_CR3) 326
11.10.4 PWR voltage scaling register (PWR_VOSR) 326
11.10.5 PWR supply voltage monitoring control register (PWR_SVMCR) 327
11.10.6 PWR wake-up control register 1 (PWR_WUCR1) 328
11.10.7 PWR wake-up control register 2 (PWR_WUCR2) 329
11.10.8 PWR wake-up control register 3 (PWR_WUCR3) 331
11.10.9 PWR disable backup domain register (PWR_DBPR) 333
11.10.10 PWR security configuration register (PWR_SECCFGR) 333
11.10.11 PWR privilege control register (PWR_PRIVCFGR) 335
11.10.12 PWR status register (PWR_SR) 335

11.10.13 PWR supply voltage monitoring status register (PWR_SVMSR) 336
11.10.14 PWR wake-up status register (PWR_WUSR) 337
11.10.15 PWR wake-up status clear register (PWR_WUSCR) 338
11.10.16 PWR port A Standby I/O retention enable register (PWR_IORETENRA) 339
11.10.17 PWR port A Standby I/O retention status register (PWR_IORETRA) 340
11.10.18 PWR port B Standby I/O retention enable register (PWR_IORETENRB) 340
11.10.19 PWR port B Standby I/O retention status register (PWR_IORETRB) 341
11.10.20 PWR port C Standby I/O retention enable register (PWR_IORETENRC) 342
11.10.21 PWR port C Standby I/O retention status register (PWR_IORETRC) 342
11.10.22 PWR port H Standby I/O retention enable register (PWR_IORETENRH) 343
11.10.23 PWR port H Standby I/O retention status register (PWR_IORETRH) 344
11.10.24 PWR 2.4 GHz RADIO status and control register (PWR_RADIOSCR) 344
11.10.25 PWR Stop 2 peripheral IOs retention register (PWR_S2RETR) 346
11.10.26 PWR register map 347

12 Reset and clock control (RCC) 350

12.1 Introduction 350
12.2 RCC pins and internal signals 350
12.3 RCC reset functional description 350
12.3.1 Power reset 350
12.3.2 System reset 351
12.3.3 Backup domain reset 352
12.3.4 Individual peripheral reset 352
12.3.5 CPU reset 352
12.4 RCC clocks functional description 352
12.4.1 HSE32 clock with trimming 354
12.4.2 HSI16 clock 356
12.4.3 PLL1 357
12.4.4 LSE clock 359
12.4.5 LSI clock 360

12.4.6 System clock (SYSCLK) selection . . . 362
12.4.7 Clock source frequency versus voltage scaling . . . 365
12.4.8 HSE32 clock security system (HSECSS) . . . 365
12.4.9 LSE clock security system on (LSECSS) . . . 365
12.4.10 ADC kernel clock . . . 366
12.4.11 RTC and TAMP kernel clock . . . 366
12.4.12 2.4 GHz RADIO bus clocks . . . 367
12.4.13 2.4 GHz RADIO kernel clocks . . . 368
12.4.14 Timer kernel clock . . . 368
12.4.15 Independent watchdog kernel clock . . . 369
12.4.16 USB kernel clock . . . 369
12.4.17 XSPI feedback clock delay . . . 369
12.4.18 SysTick calibration value register . . . 369
12.4.19 Clock-out capability . . . 370
12.4.20 Internal/external clock measurement . . . 370
12.4.21 Audio synchronization . . . 371
12.4.22 Peripheral clock gating and autonomous mode . . . 373
12.5 RCC security and privilege functional description . . . 376
12.5.1 RCC TrustZone® security protection modes . . . 376
12.5.2 RCC privilege protection modes . . . 378
12.6 RCC low-power modes . . . 379
12.7 RCC interrupts . . . 380
12.8 RCC registers . . . 381
12.8.1 RCC clock control register (RCC_CR) . . . 381
12.8.2 RCC internal clock sources calibration register 3 (RCC_ICSCR3) . . . 383
12.8.3 RCC clock configuration register 1 (RCC_CFGR1) . . . 384
12.8.4 RCC clock configuration register 2 (RCC_CFGR2) . . . 385
12.8.5 RCC clock configuration register 3 (RCC_CFGR3) . . . 386
12.8.6 RCC PLL1 configuration register (RCC_PLL1CFGR) . . . 387
12.8.7 RCC PLL1 dividers register (RCC_PLL1DIVR) . . . 389
12.8.8 RCC PLL1 fractional divider register (RCC_PLL1FRACR) . . . 390
12.8.9 RCC clock interrupt enable register (RCC_CIER) . . . 391
12.8.10 RCC clock interrupt flag register (RCC_CIFR) . . . 393
12.8.11 RCC clock interrupt clear register (RCC_CICR) . . . 395
12.8.12 RCC AHB1 peripheral reset register (RCC_AHB1RSTR) . . . 396
12.8.13 RCC AHB2 peripheral reset register (RCC_AHB2RSTR) . . . 397
12.8.14 RCC AHB4 peripheral reset register (RCC_AHB4RSTR) . . . 399

12.8.15 RCC AHB5 peripheral reset register (RCC_AHB5RSTR) 400
12.8.16 RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1) 400
12.8.17 RCC APB1 peripheral reset register 2 (RCC_APB1RSTR2) 401
12.8.18 RCC APB2 peripheral reset register (RCC_APB2RSTR) 402
12.8.19 RCC APB7 peripheral reset register (RCC_APB7RSTR) 403
12.8.20 RCC AHB1 peripheral clock enable register (RCC_AHB1ENR) 405
12.8.21 RCC AHB2 peripheral clock enable register (RCC_AHB2ENR) 406
12.8.22 RCC AHB4 peripheral clock enable register (RCC_AHB4ENR) 409
12.8.23 RCC AHB5 peripheral clock enable register (RCC_AHB5ENR) 410
12.8.24 RCC APB1 peripheral clock enable register 1 (RCC_APB1ENR1) . . . 411
12.8.25 RCC APB1 peripheral clock enable register 2 (RCC_APB1ENR2) . . . 412
12.8.26 RCC APB2 peripheral clock enable register (RCC_APB2ENR) 412
12.8.27 RCC APB7 peripheral clock enable register (RCC_APB7ENR) 414
12.8.28 RCC AHB1 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB1SMENR) 415
12.8.29 RCC AHB2 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB2SMENR) 417
12.8.30 RCC AHB4 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB4SMENR) 420
12.8.31 RCC AHB5 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB5SMENR) 421
12.8.32 RCC APB1 peripheral clocks enable in Sleep and Stop modes register 1 (RCC_APB1SMENR1) 422
12.8.33 RCC APB1 peripheral clocks enable in Sleep and Stop modes register 2 (RCC_APB1SMENR2) 423
12.8.34 RCC APB2 peripheral clocks enable in Sleep and Stop modes register (RCC_APB2SMENR) 423
12.8.35 RCC APB7 peripheral clock enable in Sleep and Stop modes register (RCC_APB7SMENR) 425
12.8.36 RCC peripherals independent clock configuration register 1 (RCC_CCIPR1) 427
12.8.37 RCC peripherals independent clock configuration register 2 (RCC_CCIPR2) 429
12.8.38 RCC peripherals independent clock configuration register 3 (RCC_CCIPR3) 430
12.8.39 RCC backup domain control register (RCC_BDCR1) 431
12.8.40 RCC Backup domain control register (RCC_BDCR2) 437
12.8.41 RCC control/status register (RCC_CSR) 438
12.8.42 RCC secure configuration register (RCC_SECCFGR) 439
12.8.43 RCC privilege configuration register (RCC_PRIVCFGR) 441

12.8.44 RCC audio synchronization control register (RCC_ASCR) 441
12.8.45 RCC audio synchronization interrupt enable register (RCC_ASIER) . . 442
12.8.46 RCC audio synchronization status register (RCC_ASSR) 443
12.8.47 RCC audio synchronization counter register (RCC_ASCNTR) 444
12.8.48 RCC audio synchronization auto-reautoreload register (RCC_ASARR) 445
12.8.49 RCC audio synchronization capture register (RCC_ASCAR) 445
12.8.50 RCC audio synchronization compare register (RCC_ASCOR) 446
12.8.51 RCC clock configuration register 2 (RCC_CFGR4) 446
12.8.52 RCC RADIO peripheral clock enable register (RCC_RADIOENR) . . . 447
12.8.53 RCC external clock sources calibration register 1(RCC_ECSCR1) . . . 448
12.8.54 RCC LSI key register (RCC_LSIKEYR) 449
12.8.55 RCC register map 451

13 General-purpose I/Os (GPIO) 458

13.1 GPIO introduction 458
13.2 GPIO main features 458
13.3 GPIO implementation 458
13.4 GPIO functional description 459
13.4.1 GPIO general-purpose I/O 461
13.4.2 GPIO pin alternate function multiplexer and mapping 461
13.4.3 GPIO port control registers 462
13.4.4 GPIO port data registers 462
13.4.5 GPIO data bitwise handling 462
13.4.6 GPIO locking mechanism 463
13.4.7 GPIO alternate function input/output 463
13.4.8 GPIO external interrupt/wake-up lines 463
13.4.9 GPIO input configuration 464
13.4.10 GPIO output configuration 464
13.4.11 GPIO alternate function configuration 465
13.4.12 GPIO analog configuration 466
13.4.13 GPIO compensation cell 466
13.4.14 GPIO Standby retention 466
13.4.15 GPIO using the LSE oscillator pins as GPIOs 466
13.4.16 GPIO using GPIO pins with RTC 466
13.4.17 GPIO using PH3 as GPIO 467
13.4.18 GPIO TrustZone® security 467

13.4.19 GPIO privileged and unprivileged modes 468
13.5 GPIO port A registers 469
13.5.1 GPIO port A mode register (GPIOA_MODER) 469
13.5.2 GPIO port A output type register (GPIOA_OTYPER) 470
13.5.3 GPIO port A output speed register (GPIOA_OSPEEDR) 470
13.5.4 GPIO port A pull-up/pull-down register (GPIOA_PUPDR) 471
13.5.5 GPIO port A input data register (GPIOA_IDR) 472
13.5.6 GPIO port A output data register (GPIOA_ODR) 472
13.5.7 GPIO port A bit set/reset register (GPIOA_BSRR) 473
13.5.8 GPIO port A configuration lock register (GPIOA_LCKR) 473
13.5.9 GPIO port A alternate function low register (GPIOA_AFRL) 474
13.5.10 GPIO port A alternate function high register (GPIOA_AFRH) 475
13.5.11 GPIO port A bit reset register (GPIOA_BRR) 476
13.5.12 GPIO port A secure configuration register (GPIOA_SECCFGR) 477
13.6 GPIO port B registers 477
13.6.1 GPIO port B mode register (GPIOB_MODER) 477
13.6.2 GPIO port B output type register (GPIOB_OTYPER) 478
13.6.3 GPIO port B output speed register (GPIOB_OSPEEDR) 479
13.6.4 GPIO port B pull-up/pull-down register (GPIOB_PUPDR) 480
13.6.5 GPIO port B input data register (GPIOB_IDR) 480
13.6.6 GPIO port B output data register (GPIOB_ODR) 481
13.6.7 GPIO port B bit set/reset register (GPIOB_BSRR) 482
13.6.8 GPIO port B configuration lock register (GPIOB_LCKR) 482
13.6.9 GPIO port B alternate function low register (GPIOB_AFRL) 484
13.6.10 GPIO port B alternate function high register (GPIOB_AFRH) 485
13.6.11 GPIO port B bit reset register (GPIOB_BRR) 485
13.6.12 GPIO port B secure configuration register (GPIOB_SECCFGR) 486
13.7 GPIO port C registers 487
13.7.1 GPIO port C mode register (GPIOC_MODER) 487
13.7.2 GPIO port C output type register (GPIOC_OTYPER) 488
13.7.3 GPIOC port output speed register (GPIOC_OSPEEDR) 488
13.7.4 GPIO port C pull-up/pull-down register (GPIOC_PUPDR) 489
13.7.5 GPIO port C input data register (GPIOC_IDR) 489
13.7.6 GPIO port C output data register (GPIOC_ODR) 490
13.7.7 GPIO port C bit set/reset register (GPIOC_BSRR) 490
13.7.8 GPIO port C configuration lock register (GPIOC_LCKR) 491
13.7.9 GPIO port C alternate function high register (GPIOC_AFRH) 492

13.7.10 GPIO port C bit reset register (GPIOC_BRR) 492
13.7.11 GPIO port C secure configuration register (GPIOC_SECCFGR) 493
13.8 GPIO port H registers 493
13.8.1 GPIO port H mode register (GPIOH_MODER) 493
13.8.2 GPIO port H output type register (GPIOH_OTYPER) 494
13.8.3 GPIO port H output speed register (GPIOH_OSPEEDR) 494
13.8.4 GPIO port H pull-up/pull-down register (GPIOH_PUPDR) 495
13.8.5 GPIO port H input data register (GPIOH_IDR) 495
13.8.6 GPIO port H output data register (GPIOH_ODR) 496
13.8.7 GPIO port H bit set/reset register (GPIOH_BSRR) 496
13.8.8 GPIO port H configuration lock register (GPIOH_LCKR) 497
13.8.9 GPIO port H alternate function low register (GPIOH_AFRL) 498
13.8.10 GPIO port H bit reset register (GPIOH_BRR) 499
13.8.11 GPIO port H secure configuration register (GPIOH_SECCFGR) 499
13.8.12 GPIOA register map 500
13.8.13 GPIOB register map 501
13.8.14 GPIOC register map 502
13.8.15 GPIOH register map 503

14 System configuration controller (SYSCFG) 504
14.1 SYSCFG main features 504
14.2 SYSCFG functional description 504
14.2.1 I/O compensation cell management 504
14.2.2 SYSCFG TrustZone® security and privilege 505
14.3 SYSCFG registers 507
14.3.1 SYSCFG secure configuration register (SYSCFG_SECCFGR) 507
14.3.2 SYSCFG configuration register 1 (SYSCFG_CFGR1) 507
14.3.3 SYSCFG FPU interrupt mask register (SYSCFG_FPUIMR) 510
14.3.4 SYSCFG CPU nonsecure lock register (SYSCFG_CNSLCKR) 510
14.3.5 SYSCFG CPU secure lock register (SYSCFG_CSLOCKR) 511
14.3.6 SYSCFG configuration register 2 (SYSCFG_CFGR2) 512
14.3.7 SYSCFG memory erase status register (SYSCFG_MESR) 513
14.3.8 SYSCFG compensation cell control/status register (SYSCFG_CCCSR) 514
14.3.9 SYSCFG compensation cell value register (SYSCFG_CCVR) 515
14.3.10 SYSCFG compensation cell code register (SYSCFG_CCCR) 516
14.3.11 SYSCFG flash abort interrupt enable register . . . .(SYSCFG_FAIER) 516

14.3.12 SYSCFG register map 520

15 Peripherals interconnect matrix 521

15.1 Introduction 521

15.2 Connection summary 521

15.3 Interconnection details 522

15.3.1 Master to slave interconnection for timers 522

15.3.2 Triggers to ADC4 522

15.3.3 ADC4 analog watchdog as trigger to timer 523

15.3.4 Clock source to timer 523

15.3.5 Triggers to low-power timer 524

15.3.6 Internal analog signals to analog peripheral 525

15.3.7 System errors as break signals to timers 525

15.3.8 Triggers to LPDMA1 525

15.3.9 Internal tamper sources 526

15.3.10 Triggers to communication peripherals 526

15.3.11 Output from tamper 527

15.3.12 Timers generating IRTIM signal 527

16 Low-power direct memory access controller (LPDMA) 528

16.1 LPDMA introduction 528

16.2 LPDMA main features 528

16.3 LPDMA implementation 529

16.3.1 LPDMA channels 529

16.3.2 LPDMA autonomous mode in low-power modes 529

16.3.3 LPDMA requests 530

16.3.4 LPDMA block requests 531

16.3.5 LPDMA triggers 531

16.4 LPDMA functional description 533

16.4.1 LPDMA block diagram 533

16.4.2 LPDMA channel state and direct programming without any linked-list 533

16.4.3 LPDMA channel suspend and resume 534

16.4.4 LPDMA channel abort and restart 535

16.4.5 LPDMA linked-list data structure 536

16.4.6 Linked-list item transfer execution 538

16.4.7 LPDMA channel state and linked-list programming in run-to-completion mode . . . 538
16.4.8 LPDMA channel state and linked-list programming in link step mode . 543
16.4.9 LPDMA channel state and linked-list programming . . . 549
16.4.10 LPDMA direct transfers . . . 551
16.4.11 LPDMA transfer request and arbitration . . . 553
16.4.12 LPDMA triggered transfer . . . 557
16.4.13 LPDMA circular buffering with linked-list programming . . . 558
16.4.14 LPDMA secure/nonsecure channel . . . 560
16.4.15 LPDMA privileged/unprivileged channel . . . 561
16.4.16 LPDMA error management . . . 562
16.4.17 LPDMA autonomous mode . . . 563
16.5 LPDMA in debug mode . . . 564
16.6 LPDMA in low-power modes . . . 564
16.7 LPDMA interrupts . . . 565
16.8 LPDMA registers . . . 566
16.8.1 LPDMA secure configuration register (LPDMA_SECCFGR) . . . 566
16.8.2 LPDMA privileged configuration register (LPDMA_PRIVCFGR) . . . 567
16.8.3 LPDMA configuration lock register (LPDMA_RCFGLOCKR) . . . 568
16.8.4 LPDMA nonsecure masked interrupt status register (LPDMA_MISR) . . . 568
16.8.5 LPDMA secure masked interrupt status register (LPDMA_SMISR) . . . 569
16.8.6 LPDMA channel x linked-list base address register (LPDMA_CxLBAR) . . . 570
16.8.7 LPDMA channel x flag clear register (LPDMA_CxFCR) . . . 570
16.8.8 LPDMA channel x status register (LPDMA_CxSR) . . . 571
16.8.9 LPDMA channel x control register (LPDMA_CxCR) . . . 573
16.8.10 LPDMA channel x transfer register 1 (LPDMA_CxTR1) . . . 575
16.8.11 LPDMA channel x transfer register 2 (LPDMA_CxTR2) . . . 577
16.8.12 LPDMA channel x block register 1 (LPDMA_CxBR1) . . . 580
16.8.13 LPDMA channel x source address register (LPDMA_CxSAR) . . . 581
16.8.14 LPDMA channel x destination address register (LPDMA_CxDAR) . . . 582
16.8.15 LPDMA channel x linked-list address register (LPDMA_CxLLR) . . . 583
16.8.16 LPDMA register map . . . 584

17 Nested vectored interrupt controller (NVIC) . . . 586

17.1 NVIC main features . . . 586

17.2 Interrupt and exception vectors 586

18 Extended interrupts and event controller (EXTI) 590

18.1 EXTI main features 590

18.2 EXTI block diagram 590

18.2.1 EXTI connections between peripherals and CPU 592

18.2.2 EXTI interrupt/event mapping 592

18.3 EXTI functional description 592

18.3.1 EXTI configurable event input wake-up 593

18.3.2 EXTI mux selection 593

18.4 EXTI functional behavior 594

18.5 EXTI event protection 595

18.5.1 EXTI security protection 595

18.5.2 EXTI privilege protection 596

18.6 EXTI registers 597

18.6.1 EXTI rising trigger selection register (EXTI_RTSR1) 597

18.6.2 EXTI falling trigger selection register (EXTI_FTSR1) 598

18.6.3 EXTI software interrupt event register (EXTI_SWIER1) 598

18.6.4 EXTI rising edge pending register (EXTI_RPR1) 599

18.6.5 EXTI falling edge pending register (EXTI_FPR1) 600

18.6.6 EXTI security configuration register (EXTI_SECCFGR1) 600

18.6.7 EXTI privilege configuration register (EXTI_PRIVCFGR1) 601

18.6.8 EXTI external interrupt selection register (EXTI_EXTICR1) 601

18.6.9 EXTI external interrupt selection register (EXTI_EXTICR2) 603

18.6.10 EXTI external interrupt selection register (EXTI_EXTICR3) 604

18.6.11 EXTI external interrupt selection register (EXTI_EXTICR4) 605

18.6.12 EXTI lock register (EXTI_LOCKR) 607

18.6.13 EXTI CPU wake-up with interrupt mask register (EXTI_IMR1) 607

18.6.14 EXTI CPU wake-up with event mask register (EXTI_EMR1) 608

18.6.15 EXTI register map 609

19 Cyclic redundancy check calculation unit (CRC) 611

19.1 CRC introduction 611

19.2 CRC main features 611

19.3 CRC functional description 612

19.3.1 CRC block diagram 612

19.3.2 CRC internal signals 612
19.3.3 CRC operation 612
19.4 CRC registers 614
19.4.1 CRC data register (CRC_DR) 614
19.4.2 CRC independent data register (CRC_IDR) 614
19.4.3 CRC control register (CRC_CR) 615
19.4.4 CRC initial value (CRC_INIT) 616
19.4.5 CRC polynomial (CRC_POL) 616
19.4.6 CRC register map 617

20 Extended-SPI interface (XSPI) 618
20.1 XSPI introduction 618
20.2 XSPI main features 618
20.3 XSPI implementation 618
20.4 XSPI functional description 620
20.4.1 XSPI block diagram 620
20.4.2 XSPI pins and internal signals 620
20.4.3 XSPI interface to memory modes 621
20.4.4 XSPI regular-command protocol 621
20.4.5 XSPI regular-command protocol signal interface 624
20.4.6 XSPI operating mode introduction 628
20.4.7 XSPI indirect mode 628
20.4.8 XSPI memory-mapped mode 629
20.4.9 XSPI configuration introduction 630
20.4.10 XSPI system configuration 630
20.4.11 XSPI device configuration 631
20.4.12 XSPI regular-command mode configuration 633
20.4.13 XSPI error management 635
20.4.14 XSPI BUSY and ABORT 635
20.4.15 XSPI reconfiguration or deactivation 635
20.4.16 NCS behavior 636
20.4.17 Software control of two external memories 636
20.5 Address alignment and data number 637
20.6 XSPI interrupts 638
20.7 XSPI registers 638
20.7.1 XSPI control register (XSPI_CR) 638

20.7.2 XSPI device configuration register 1 (XSPI_DCR1) 641
20.7.3 XSPI device configuration register 2 (XSPI_DCR2) 643
20.7.4 XSPI status register (XSPI_SR) 644
20.7.5 XSPI flag clear register (XSPI_FCR) 645
20.7.6 XSPI data length register (XSPI_DLR) 645
20.7.7 XSPI address register (XSPI_AR) 646
20.7.8 XSPI data register (XSPI_DR) 646
20.7.9 XSPI communication configuration register (XSPI_CCR) 647
20.7.10 XSPI timing configuration register (XSPI_TCR) 649
20.7.11 XSPI instruction register (XSPI_IR) 650
20.7.12 XSPI alternate bytes register (XSPI_ABR) 650
20.7.13 XSPI low-power timeout register (XSPI_LPTR) 650
20.7.14 XSPI register map 651

21 Delay block (DLYB) 653

21.1 DLYB introduction 653
21.2 DLYB main features 653
21.3 DLYB implementation 653
21.4 DLYB functional description 653
21.4.1 DLYB diagram 653
21.4.2 DLYB pins and internal signals 654
21.4.3 General description 654
21.4.4 Delay line length configuration procedure 655
21.4.5 Output clock phase configuration procedure 656
21.5 DLYB registers 657
21.5.1 DLYB control register (DLYB_CR) 657
21.5.2 DLYB configuration register (DLYB_CFGR) 657
21.5.3 DLYB register map 658

22 Analog-to-digital converter (ADC4) 659

22.1 ADC introduction 659
22.2 ADC main features 659
22.3 ADC implementation 660
22.4 ADC functional description 662
22.4.1 ADC block diagram 662
22.4.2 ADC pins and internal signals 662

22.4.3 ADC voltage regulator (ADVREGEN) 663
22.4.4 Calibration (ADCAL) 664
22.4.5 ADC on-off control (ADEN, ADDIS, ADRDY) 666
22.4.6 ADC clock (PRESC[3:0]) 668
22.4.7 ADC connectivity 669
22.4.8 Configuring the ADC 670
22.4.9 Channel selection (CHSEL, SCANDIR, CHSELRMOD) 670
22.4.10 Programmable sampling time (SMPx[2:0]) 671
22.4.11 Single conversion mode (CONT = 0) 672
22.4.12 Continuous conversion mode (CONT = 1) 672
22.4.13 Starting conversions (ADSTART) 673
22.4.14 Timings 674
22.4.15 Stopping an ongoing conversion (ADSTP) 675
22.4.16 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN) 675
22.4.17 Discontinuous mode (DISCEN) 676
22.4.18 Programmable resolution (RES) - fast conversion mode 676
22.4.19 End of conversion, end of sampling phase (EOC, EOSMP flags) 677
22.4.20 End of conversion sequence (EOS flag) 678
22.4.21 Example timing diagrams (single/continuous modes hardware/software triggers) 678
22.4.22 Low-frequency trigger mode 680
22.4.23 Data management 680
22.4.24 Low-power features 684
22.4.25 Analog window watchdog 688
22.4.26 Oversampler 692
22.4.27 Temperature sensor and internal reference voltage 695
22.5 ADC in low-power modes 698
22.6 ADC interrupts 698
22.7 ADC registers 700
22.7.1 ADC interrupt and status register (ADC_ISR) 700
22.7.2 ADC interrupt enable register (ADC_IER) 701
22.7.3 ADC control register (ADC_CR) 704
22.7.4 ADC configuration register 1 (ADC_CFGR1) 706
22.7.5 ADC configuration register 2 (ADC_CFGR2) 709
22.7.6 ADC sampling time register (ADC_SMPR) 710
22.7.7 ADC watchdog threshold register (ADC_AWD1TR) 711

22.7.8 ADC watchdog threshold register (ADC_AWD2TR) 712
22.7.9 ADC channel selection register [alternate] (ADC_CHSELR) 713
22.7.10 ADC channel selection register [alternate] (ADC_CHSELR) 713
22.7.11 ADC watchdog threshold register (ADC_AWD3TR) 715
22.7.12 ADC data register (ADC_DR) 716
22.7.13 ADC power register (ADC_PWRR) 716
22.7.14 ADC Analog Watchdog 2 Configuration register (ADC_AWD2CR) . . . 717
22.7.15 ADC Analog Watchdog 3 Configuration register (ADC_AWD3CR) . . . 717
22.7.16 ADC Calibration factor (ADC_CALFACT) 718
22.7.17 ADC common configuration register (ADC_CCR) 718
22.7.18 ADC register map 719

23 True random number generator (RNG) 722

23.1 RNG introduction 722
23.2 RNG main features 722
23.3 RNG functional description 723
23.3.1 RNG block diagram 723
23.3.2 RNG internal signals 723
23.3.3 Random number generation 724
23.3.4 RNG initialization 726
23.3.5 RNG operation 727
23.3.6 RNG clocking 729
23.3.7 Error management 729
23.3.8 RNG low-power use 731
23.4 RNG interrupts 731
23.5 RNG processing time 732
23.6 RNG entropy source validation 732
23.6.1 Introduction 732
23.6.2 Validation conditions 732
23.7 RNG registers 734
23.7.1 RNG control register (RNG_CR) 734
23.7.2 RNG status register (RNG_SR) 736
23.7.3 RNG data register (RNG_DR) 737
23.7.4 RNG noise source control register (RNG_NSCR) 738
23.7.5 RNG health test control register x (RNG_HTCRx) 738
23.7.6 RNG health test status register 0 (RNG_HTSR0) 739

23.7.7 RNG health test status register 1 (RNG_HTSR1) 739
23.7.8 RNG noise source mask register (RNG_NSMR) 740
23.7.9 RNG register map 741

24 AES hardware accelerator (AES) 742
24.1 AES introduction 742
24.2 AES main features 742
24.3 AES implementation 742
24.4 AES functional description 743
24.4.1 AES block diagram 743
24.4.2 AES internal signals 743
24.4.3 AES reset and clocks 744
24.4.4 AES symmetric cipher implementation 744
24.4.5 AES encryption or decryption typical usage 745
24.4.6 AES authenticated encryption, decryption, and cipher-based message authentication 747
24.4.7 AES ciphertext stealing and data padding 747
24.4.8 AES suspend and resume operations 748
24.4.9 AES basic chaining modes (ECB, CBC) 749
24.4.10 AES counter (CTR) mode 752
24.4.11 AES Galois/counter mode (GCM) 754
24.4.12 AES Galois message authentication code (GMAC) 759
24.4.13 AES counter with CBC-MAC (CCM) 760
24.4.14 AES data registers and data swapping 765
24.4.15 AES key registers 768
24.4.16 AES initialization vector registers 768
24.4.17 AES error management 769
24.5 AES in low-power modes 769
24.6 AES interrupts 770
24.7 AES DMA requests 770
24.8 AES processing latency 771
24.9 AES registers 772
24.9.1 AES control register (AES_CR) 772
24.9.2 AES status register (AES_SR) 774
24.9.3 AES data input register (AES_DINR) 775
24.9.4 AES data output register (AES_DOUTR) 775

24.9.5 AES key register 0 (AES_KEYR0) 776
24.9.6 AES key register 1 (AES_KEYR1) 776
24.9.7 AES key register 2 (AES_KEYR2) 777
24.9.8 AES key register 3 (AES_KEYR3) 777
24.9.9 AES initialization vector register 0 (AES_IVR0) 777
24.9.10 AES initialization vector register 1 (AES_IVR1) 778
24.9.11 AES initialization vector register 2 (AES_IVR2) 778
24.9.12 AES initialization vector register 3 (AES_IVR3) 778
24.9.13 AES key register 4 (AES_KEYR4) 779
24.9.14 AES key register 5 (AES_KEYR5) 779
24.9.15 AES key register 6 (AES_KEYR6) 779
24.9.16 AES key register 7 (AES_KEYR7) 780
24.9.17 AES suspend registers (AES_SUSPRx) 780
24.9.18 AES interrupt enable register (AES_IER) 780
24.9.19 AES interrupt status register (AES_ISR) 781
24.9.20 AES interrupt clear register (AES_ICR) 782
24.9.21 AES register map 783

25 Hash processor (HASH) 785

25.1 HASH introduction 785
25.2 HASH main features 785
25.3 HASH implementation 786
25.4 HASH functional description 786
25.4.1 HASH block diagram 786
25.4.2 HASH internal signals 786
25.4.3 About secure hash algorithms 787
25.4.4 Message data feeding 787
25.4.5 Message digest computing 788
25.4.6 Message padding 790
25.4.7 HMAC operation 791
25.4.8 HASH suspend/resume operations 793
25.4.9 HASH DMA interface 795
25.4.10 HASH error management 795
25.4.11 HASH processing time 795
25.5 HASH interrupts 796
25.6 HASH registers 796

25.6.1 HASH control register (HASH_CR) 796
25.6.2 HASH data input register (HASH_DIN) 798
25.6.3 HASH start register (HASH_STR) 799
25.6.4 HASH digest registers 800
25.6.5 HASH interrupt enable register (HASH_IMR) 802
25.6.6 HASH status register (HASH_SR) 802
25.6.7 HASH context swap registers 803
25.6.8 HASH register map 804

26 On-the-fly decryption engine (OTFDEC) 806

26.1 OTFDEC introduction 806
26.2 OTFDEC main features 806
26.3 OTFDEC functional description 807
26.3.1 OTFDEC block diagram 807
26.3.2 OTFDEC internal signals 807
26.3.3 OTFDEC on-the-fly decryption 808
26.3.4 OTFDEC usage of AES in counter mode decryption 809
26.3.5 Flow control management 810
26.3.6 OTFDEC error management 810
26.4 OTFDEC interrupts 811
26.5 OTFDEC application information 811
26.5.1 OTFDEC initialization process 811
26.5.2 OTFDEC and power management 813
26.5.3 Encrypting for OTFDEC 813
26.5.4 OTFDEC key CRC source code 814
26.6 OTFDEC registers 815
26.6.1 OTFDEC control register (OTFDEC_CR) 815
26.6.2 OTFDEC privileged access control configuration register (OTFDEC_PRIVCFGR) 816
26.6.3 OTFDEC region x configuration register (OTFDEC_RxCFGR) 816
26.6.4 OTFDEC region x start address register (OTFDEC_RxSTARTADDR) 818
26.6.5 OTFDEC region x end address register (OTFDEC_RxENDADDR) . . 818
26.6.6 OTFDEC region x nonce register 0 (OTFDEC_RxNONCER0) 819
26.6.7 OTFDEC region x nonce register 1 (OTFDEC_RxNONCER1) 820
26.6.8 OTFDEC region x key register 0 (OTFDEC_RxKEYR0) 820
26.6.9 OTFDEC region x key register 1 (OTFDEC_RxKEYR1) 821

26.6.10 OTFDEC region x key register 2 (OTFDEC_RxKEYR2) 821
26.6.11 OTFDEC region x key register 3 (OTFDEC_RxKEYR3) 822
26.6.12 OTFDEC interrupt status register (OTFDEC_ISR) 822
26.6.13 OTFDEC interrupt clear register (OTFDEC_ICR) 823
26.6.14 OTFDEC interrupt enable register (OTFDEC_IER) 824
26.6.15 OTFDEC register map 825

27 Public key accelerator (PKA) 829

27.1 PKA introduction 829
27.2 PKA main features 829
27.3 PKA functional description 830
27.3.1 PKA block diagram 830
27.3.2 PKA internal signals 830
27.3.3 PKA reset and clocks 830
27.3.4 PKA public key acceleration 831
27.3.5 Typical applications for PKA 833
27.3.6 PKA procedure to perform an operation 835
27.3.7 PKA error management 836
27.4 PKA operating modes 837
27.4.1 Introduction 837
27.4.2 Montgomery parameter computation 838
27.4.3 Modular addition 838
27.4.4 Modular subtraction 839
27.4.5 Modular and Montgomery multiplication 839
27.4.6 Modular exponentiation 840
27.4.7 Modular inversion 842
27.4.8 Modular reduction 842
27.4.9 Arithmetic addition 843
27.4.10 Arithmetic subtraction 843
27.4.11 Arithmetic multiplication 844
27.4.12 Arithmetic comparison 844
27.4.13 RSA CRT exponentiation 844
27.4.14 Point on elliptic curve Fp check 845
27.4.15 ECC Fp scalar multiplication 846
27.4.16 ECDSA sign 847
27.4.17 ECDSA verification 849
27.4.18 ECC complete addition 850

27.4.19 ECC double base ladder 850
27.4.20 ECC projective to affine 851
27.5 Example of configurations and processing times 852
27.5.1 Supported elliptic curves 852
27.5.2 Computation times 854
27.6 PKA in low-power modes 856
27.7 PKA interrupts 856
27.8 PKA registers 857
27.8.1 PKA control register (PKA_CR) 857
27.8.2 PKA status register (PKA_SR) 859
27.8.3 PKA clear flag register (PKA_CLRFR) 860
27.8.4 PKA RAM 860
27.8.5 PKA register map 861

28 General-purpose timer (TIM2) 862
28.1 TIM2 introduction 862
28.2 TIM2 main features 862
28.3 TIM2 implementation 863
28.4 TIM2 functional description 864
28.4.1 Block diagram 864
28.4.2 TIM2 pins and internal signals 865
28.4.3 Time-base unit 868
28.4.4 Counter modes 870
28.4.5 Clock selection 881
28.4.6 Capture/compare channels 885
28.4.7 Input capture mode 887
28.4.8 PWM input mode 888
28.4.9 Forced output mode 889
28.4.10 Output compare mode 889
28.4.11 PWM mode 891
28.4.12 Asymmetric PWM mode 899
28.4.13 Combined PWM mode 900
28.4.14 Clearing the tim_ocxref signal on an external event 901
28.4.15 One-pulse mode 903
28.4.16 Retriggerable one-pulse mode 904
28.4.17 Pulse on compare mode 905

28.4.18 Encoder interface mode 907
28.4.19 Direction bit output 925
28.4.20 UIF bit remapping 926
28.4.21 Timer input XOR function 926
28.4.22 Timers and external trigger synchronization 926
28.4.23 Timer synchronization 930
28.4.24 ADC triggers 935
28.4.25 ADC synchronization 936
28.4.26 DMA burst mode 937
28.4.27 TIM2 DMA requests 938
28.4.28 Debug mode 938
28.4.29 TIM2 low-power modes 938
28.4.30 TIM2 interrupts 939
28.5 TIM2 registers 940
28.5.1 TIM2 control register 1 (TIM2_CR1) 940
28.5.2 TIM2 control register 2 (TIM2_CR2) 941
28.5.3 TIM2 slave mode control register (TIM2_SMCR) 943
28.5.4 TIM2 DMA/Interrupt enable register (TIM2_DIER) 947
28.5.5 TIM2 status register (TIM2_SR) 948
28.5.6 TIM2 event generation register (TIM2_EGR) 950
28.5.7 TIM2 capture/compare mode register 1 (TIM2_CCMR1) 951
28.5.8 TIM2 capture/compare mode register 1 [alternate] (TIM2_CCMR1) 953
28.5.9 TIM2 capture/compare mode register 2 (TIM2_CCMR2) 955
28.5.10 TIM2 capture/compare mode register 2 [alternate] (TIM2_CCMR2) 956
28.5.11 TIM2 capture/compare enable register (TIM2_CCER) 959
28.5.12 TIM2 counter (TIM2_CNT) 961
28.5.13 TIM2 prescaler (TIM2_PSC) 961
28.5.14 TIM2 autoreload register (TIM2_ARR) 962
28.5.15 TIM2 capture/compare register 1 (TIM2_CCR1) 962
28.5.16 TIM2 capture/compare register 2 (TIM2_CCR2) 963
28.5.17 TIM2 capture/compare register 3 (TIM2_CCR3) 964
28.5.18 TIM2 capture/compare register 4 (TIM2_CCR4) 965
28.5.19 TIM2 timer encoder control register (TIM2_ECR) 966
28.5.20 TIM2 timer input selection register (TIM2_TISEL) 967
28.5.21 TIM2 alternate function register 1 (TIM2_AF1) 968

28.5.22 TIM2 alternate function register 2 (TIM2_AF2) 969
28.5.23 TIM2 DMA control register (TIM2_DCR) 970
28.5.24 TIM2 DMA address for full transfer (TIM2_DMAR) 971
28.5.25 TIM2 register map 972

29 General purpose timers (TIM16/TIM17) 975

29.1 TIM16/TIM17 introduction 975
29.2 TIM16/TIM17 main features 975
29.3 Implementation 975
29.4 TIM16/TIM17 functional description 976
29.4.1 Block diagram 976
29.4.2 TIM16/TIM17 pins and internal signals 976
29.4.3 Time-base unit 979
29.4.4 Counter modes 981
29.4.5 Repetition counter 985
29.4.6 Clock selection 986
29.4.7 Capture/compare channels 988
29.4.8 Input capture mode 990
29.4.9 Forced output mode 991
29.4.10 Output compare mode 991
29.4.11 PWM mode 993
29.4.12 Complementary outputs and dead-time insertion 998
29.4.13 Using the break function 1000
29.4.14 Bidirectional break input 1005
29.4.15 Clearing the tim_ocxref signal on an external event 1006
29.4.16 6-step PWM generation 1007
29.4.17 One-pulse mode 1009
29.4.18 UIF bit remapping 1010
29.4.19 Using timer output as trigger for other timers (TIM16/TIM17 only) . . 1010
29.4.20 DMA burst mode 1011
29.4.21 TIM16/TIM17 DMA requests 1012
29.4.22 Debug mode 1012
29.5 TIM16/TIM17 low-power modes 1012
29.6 TIM16/TIM17 interrupts 1012
29.7 TIM16/TIM17 registers 1014
29.7.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17) 1014

29.7.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17) 1015
29.7.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17) 1016
29.7.4 TIMx status register (TIMx_SR)(x = 16 to 17) 1017
29.7.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17) 1018
29.7.6 TIMx capture/compare mode register 1 (TIMx_CCMR1) (x = 16 to 17) 1019
29.7.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17) 1020
29.7.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) . 1022
29.7.9 TIMx counter (TIMx_CNT)(x = 16 to 17) 1025
29.7.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17) 1025
29.7.11 TIMx auto-reautoreload register (TIMx_ARR)(x = 16 to 17) 1026
29.7.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17) 1026
29.7.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17) 1027
29.7.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) 1028
29.7.15 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 16 to 17) 1031
29.7.16 TIMx input selection register (TIMx_TISEL)(x = 16 to 17) 1032
29.7.17 TIMx alternate function register 1 (TIMx_AF1)(x = 16 to 17) 1032
29.7.18 TIMx alternate function register 2 (TIMx_AF2)(x = 16 to 17) 1035
29.7.19 TIMx option register 1 (TIMx_OR1)(x = 16 to 17) 1035
29.7.20 TIMx DMA control register (TIMx_DCR)(x = 16 to 17) 1036
29.7.21 TIM16/TIM17 DMA address for full transfer (TIMx_DMAR)(x = 16 to 17) 1037
29.7.22 TIM16/TIM17 register map 1038

30 Low-power timer (LPTIM) 1040

30.1 LPTIM introduction 1040
30.2 LPTIM main features 1040
30.3 LPTIM implementation 1040
30.4 LPTIM functional description 1042
30.4.1 LPTIM block diagram 1042
30.4.2 LPTIM pins and internal signals 1042
30.4.3 LPTIM input and trigger mapping 1044
30.4.4 LPTIM reset and clocks 1045
30.4.5 Glitch filter 1045
30.4.6 Prescaler 1046
30.4.7 Trigger multiplexer 1046
30.4.8 Operating mode 1047

30.4.9 Timeout function 1049
30.4.10 Waveform generation 1049
30.4.11 Register update 1050
30.4.12 Counter mode 1051
30.4.13 Timer enable 1051
30.4.14 Timer counter reset 1052
30.4.15 Encoder mode 1052
30.4.16 Repetition counter 1054
30.4.17 Capture/compare channels 1055
30.4.18 Input capture mode 1056
30.4.19 PWM mode 1058
30.4.20 Autonomous mode 1060
30.4.21 DMA requests 1061
30.4.22 Debug mode 1062
30.5 LPTIM low-power modes 1062
30.6 LPTIM interrupts 1062
30.7 LPTIM registers 1063
30.7.1 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1, 2) 1064
30.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1, 2) 1066
30.7.3 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1, 2) 1068
30.7.4 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1, 2) 1069
30.7.5 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1, 2) 1071
30.7.6 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1, 2) 1072
30.7.7 LPTIM configuration register (LPTIM_CFGR) 1074
30.7.8 LPTIM control register (LPTIM_CR) 1077
30.7.9 LPTIM compare register 1 (LPTIM_CCR1) 1078
30.7.10 LPTIM autoreload register (LPTIM_ARR) 1078
30.7.11 LPTIM counter register (LPTIM_CNT) 1079
30.7.12 LPTIM configuration register 2 (LPTIM_CFGR2) 1079
30.7.13 LPTIM repetition register (LPTIM_RCR) 1080
30.7.14 LPTIM capture/compare mode register 1 (LPTIM_CCMR1) 1081
30.7.15 LPTIM compare register 2 (LPTIM_CCR2) 1083

30.7.16 LPTIM register map 1084

31 Infrared interface (IRTIM) 1086

32 Independent watchdog (IWDG) 1087

32.1 IWDG introduction 1087
32.2 IWDG main features 1087
32.3 IWDG implementation 1087
32.4 IWDG functional description 1088
32.4.1 IWDG block diagram 1088
32.4.2 IWDG internal signals 1089
32.4.3 Software and hardware watchdog modes 1089
32.4.4 Window option 1090
32.4.5 Debug 1093
32.4.6 Register access protection 1093
32.5 IWDG low power modes 1093
32.6 IWDG interrupts 1094
32.7 IWDG registers 1095
32.7.1 IWDG key register (IWDG_KR) 1095
32.7.2 IWDG prescaler register (IWDG_PR) 1096
32.7.3 IWDG reload register (IWDG_RLR) 1097
32.7.4 IWDG status register (IWDG_SR) 1097
32.7.5 IWDG window register (IWDG_WINR) 1098
32.7.6 IWDG early wake-up interrupt register (IWDG_EWCR) 1099
32.7.7 IWDG register map 1101

33 Real-time clock (RTC) 1102

33.1 RTC introduction 1102
33.2 RTC main features 1102
33.3 RTC functional description 1102
33.3.1 RTC block diagram 1102
33.3.2 RTC pins and internal signals 1104
33.3.3 GPIOs controlled by the RTC and TAMP 1105
33.3.4 RTC secure protection modes 1107
33.3.5 RTC privilege protection modes 1109
33.3.6 Clock and prescalers 1110

33.3.7 Real-time clock and calendar 1111
33.3.8 Calendar ultra-low power mode 1112
33.3.9 Programmable alarms 1112
33.3.10 Periodic auto-wake-up 1112
33.3.11 RTC initialization and configuration 1113
33.3.12 Reading the calendar 1116
33.3.13 Resetting the RTC 1117
33.3.14 RTC synchronization 1118
33.3.15 RTC reference clock detection 1118
33.3.16 RTC smooth digital calibration 1119
33.3.17 Timestamp function 1121
33.3.18 Calibration clock output 1122
33.3.19 Tamper and alarm output 1122
33.4 RTC low-power modes 1123
33.5 RTC interrupts 1123
33.6 RTC registers 1125
33.6.1 RTC time register (RTC_TR) 1125
33.6.2 RTC date register (RTC_DR) 1126
33.6.3 RTC subsecond register (RTC_SSR) 1127
33.6.4 RTC initialization control and status register (RTC_ICSR) 1128
33.6.5 RTC prescaler register (RTC_PRER) 1130
33.6.6 RTC wake-up timer register (RTC_WUTR) 1131
33.6.7 RTC control register (RTC_CR) 1131
33.6.8 RTC privilege mode control register (RTC_PRIVCFGR) 1135
33.6.9 RTC secure configuration register (RTC_SECCFGR) 1137
33.6.10 RTC write protection register (RTC_WPR) 1138
33.6.11 RTC calibration register (RTC_CALR) 1139
33.6.12 RTC shift control register (RTC_SHIFTR) 1140
33.6.13 RTC timestamp time register (RTC_TSTR) 1141
33.6.14 RTC timestamp date register (RTC_TSDR) 1142
33.6.15 RTC timestamp subsecond register (RTC_TSSSR) 1143
33.6.16 RTC alarm A register (RTC_ALRMAR) 1143
33.6.17 RTC alarm A subsecond register (RTC_ALRMASSR) 1145
33.6.18 RTC alarm B register (RTC_ALRMBR) 1146
33.6.19 RTC alarm B subsecond register (RTC_ALRMBSSR) 1147
33.6.20 RTC status register (RTC_SR) 1148
33.6.21 RTC nonsecure masked interrupt status register (RTC_MISR) 1149

33.6.22 RTC secure masked interrupt status register (RTC_SMISR) 1150
33.6.23 RTC status clear register (RTC_SCR) 1151
33.6.24 RTC alarm A binary mode register (RTC_ALRABINR) 1152
33.6.25 RTC alarm B binary mode register (RTC_ALRBBINR) 1153
33.6.26 RTC register map 1154

34 Tamper and backup registers (TAMP) 1156
34.1 TAMP introduction 1156
34.2 TAMP main features 1156
34.3 TAMP functional description 1157
34.3.1 TAMP block diagram 1157
34.3.2 TAMP pins and internal signals 1158
34.3.3 GPIOs controlled by the RTC and TAMP 1160
34.3.4 TAMP register write protection 1161
34.3.5 TAMP secure protection modes 1161
34.3.6 Backup registers protection zones 1162
34.3.7 TAMP privilege protection modes 1162
34.3.8 Tamper detection 1163
34.3.9 TAMP backup registers and other device secrets erase 1163
34.3.10 Tamper detection configuration and initialization 1165
34.4 TAMP low-power modes 1171
34.5 TAMP interrupts 1172
34.6 TAMP registers 1172
34.6.1 TAMP control register 1 (TAMP_CR1) 1172
34.6.2 TAMP control register 2 (TAMP_CR2) 1174
34.6.3 TAMP control register 3 (TAMP_CR3) 1176
34.6.4 TAMP filter control register (TAMP_FLTCR) 1177
34.6.5 TAMP active tamper control register 1 (TAMP_ATCR1) 1178
34.6.6 TAMP active tamper seed register (TAMP_ATSEEDR) 1181
34.6.7 TAMP active tamper output register (TAMP_ATOR) 1181
34.6.8 TAMP active tamper control register 2 (TAMP_ATCR2) 1182
34.6.9 TAMP secure configuration register (TAMP_SECCFGR) 1184
34.6.10 TAMP privilege configuration register (TAMP_PRIVCFGR) 1186
34.6.11 TAMP interrupt enable register (TAMP_IER) 1187
34.6.12 TAMP status register (TAMP_SR) 1189
34.6.13 TAMP nonsecure masked interrupt status register (TAMP_MISR) . . 1191

34.6.14 TAMP secure masked interrupt status register (TAMP_SMISR) 1192
34.6.15 TAMP status clear register (TAMP_SCR) 1194
34.6.16 TAMP monotonic counter 1 register (TAMP_COUNT1R) 1196
34.6.17 TAMP resources protection configuration register (TAMP_RPCFGR) 1196
34.6.18 TAMP backup x register (TAMP_BKPxR) 1197
34.6.19 TAMP register map 1198

35 Inter-integrated circuit interface (I2C) 1200

35.1 I2C introduction 1200
35.2 I2C main features 1200
35.3 I2C implementation 1201
35.4 I2C functional description 1201
35.4.1 I2C block diagram 1202
35.4.2 I2C pins and internal signals 1202
35.4.3 I2C clock requirements 1204
35.4.4 I2C mode selection 1204
35.4.5 I2C initialization 1205
35.4.6 I2C reset 1209
35.4.7 I2C data transfer 1210
35.4.8 I2C target mode 1212
35.4.9 I2C controller mode 1221
35.4.10 I2C_TIMINGR register configuration examples 1232
35.4.11 SMBus specific features 1234
35.4.12 SMBus initialization 1236
35.4.13 SMBus I2C_TIMEOUTR register configuration examples 1238
35.4.14 SMBus target mode 1239
35.4.15 SMBus controller mode 1242
35.4.16 Autonomous mode 1245
35.4.17 Error conditions 1247
35.5 I2C in low-power modes 1248
35.6 I2C interrupts 1249
35.7 I2C DMA requests 1249
35.7.1 Transmission using DMA 1249
35.7.2 Reception using DMA 1250
35.7.3 Controller event control using DMA 1250
35.8 I2C debug modes 1251

35.9 I2C registers 1251
35.9.1 I2C control register 1 (I2C_CR1) 1251
35.9.2 I2C control register 2 (I2C_CR2) 1254
35.9.3 I2C own address 1 register (I2C_OAR1) 1256
35.9.4 I2C own address 2 register (I2C_OAR2) 1256
35.9.5 I2C timing register (I2C_TIMINGR) 1257
35.9.6 I2C timeout register (I2C_TIMEOUTR) 1258
35.9.7 I2C interrupt and status register (I2C_ISR) 1259
35.9.8 I2C interrupt clear register (I2C_ICR) 1262
35.9.9 I2C PEC register (I2C_PECR) 1263
35.9.10 I2C receive data register (I2C_RXDR) 1263
35.9.11 I2C transmit data register (I2C_TXDR) 1264
35.9.12 I2C autonomous mode control register (I2C_AUTOCR) 1264
35.9.13 I2C register map 1266

36 Universal synchronous/asynchronous receiver transmitter (USART/UART) 1268

36.1 USART introduction 1268
36.2 USART main features 1268
36.3 USART extended features 1269
36.4 USART implementation 1269
36.5 USART functional description 1271
36.5.1 USART block diagram 1271
36.5.2 USART pins and internal signals 1271
36.5.3 USART clocks 1274
36.5.4 USART character description 1274
36.5.5 USART FIFOs and thresholds 1276
36.5.6 USART transmitter 1276
36.5.7 USART receiver 1279
36.5.8 USART baud rate generation 1286
36.5.9 Tolerance of the USART receiver to clock deviation 1288
36.5.10 USART auto baud rate detection 1289
36.5.11 USART multiprocessor communication 1291
36.5.12 USART Modbus communication 1293
36.5.13 USART parity control 1294
36.5.14 USART LIN (local interconnection network) mode 1295
36.5.15 USART synchronous mode 1297

36.5.16 USART single-wire half-duplex communication 1301
36.5.17 USART receiver timeout 1301
36.5.18 USART smartcard mode 1302
36.5.19 USART IrDA SIR ENDEC block 1306
36.5.20 Continuous communication using USART and DMA 1309
36.5.21 RS232 hardware flow control and RS485 driver enable 1311
36.5.22 USART autonomous mode 1313
36.6 USART in low-power modes 1315
36.7 USART interrupts 1316
36.8 USART registers 1318
36.8.1 USART control register 1 (USART_CR1) 1318
36.8.2 USART control register 1 [alternate] (USART_CR1) 1322
36.8.3 USART control register 2 (USART_CR2) 1325
36.8.4 USART control register 3 (USART_CR3) 1329
36.8.5 USART control register 3 [alternate] (USART_CR3) 1333
36.8.6 USART baud rate register (USART_BRR) 1336
36.8.7 USART guard time and prescaler register (USART_GTPR) 1337
36.8.8 USART receiver timeout register (USART_RTOR) 1338
36.8.9 USART request register (USART_RQR) 1339
36.8.10 USART interrupt and status register (USART_ISR) 1340
36.8.11 USART interrupt and status register [alternate] (USART_ISR) 1346
36.8.12 USART interrupt flag clear register (USART_ICR) 1351
36.8.13 USART receive data register (USART_RDR) 1352
36.8.14 USART transmit data register (USART_TDR) 1353
36.8.15 USART prescaler register (USART_PRESC) 1353
36.8.16 USART autonomous mode control register (USART_AUTOCR) 1354
36.8.17 USART register map 1355

37 Low-power universal asynchronous receiver transmitter (LPUART) 1357

37.1 LPUART introduction 1357
37.2 LPUART main features 1357
37.3 LPUART implementation 1358
37.4 LPUART functional description 1359
37.4.1 LPUART block diagram 1359
37.4.2 LPUART pins and internal signals 1360

37.4.3 LPUART clocks 1362
37.4.4 LPUART character description 1362
37.4.5 LPUART FIFOs and thresholds 1364
37.4.6 LPUART transmitter 1364
37.4.7 LPUART receiver 1368
37.4.8 LPUART baud rate generation 1372
37.4.9 Tolerance of the LPUART receiver to clock deviation 1373
37.4.10 LPUART multiprocessor communication 1374
37.4.11 LPUART parity control 1376
37.4.12 LPUART single-wire half-duplex communication 1377
37.4.13 Continuous communication using DMA and LPUART 1377
37.4.14 RS232 hardware flow control and RS485 driver enable 1380
37.4.15 LPUART autonomous mode 1382
37.5 LPUART in low-power modes 1384
37.6 LPUART interrupts 1385
37.7 LPUART registers 1386
37.7.1 LPUART control register 1 (LPUART_CR1) 1386
37.7.2 LPUART control register 1 [alternate] (LPUART_CR1) 1389
37.7.3 LPUART control register 2 (LPUART_CR2) 1392
37.7.4 LPUART control register 3 (LPUART_CR3) 1394
37.7.5 LPUART control register 3 [alternate] (LPUART_CR3) 1396
37.7.6 LPUART baud rate register (LPUART_BRR) 1398
37.7.7 LPUART request register (LPUART_RQR) 1398
37.7.8 LPUART interrupt and status register (LPUART_ISR) 1399
37.7.9 LPUART interrupt and status register [alternate] (LPUART_ISR) . . . 1404
37.7.10 LPUART interrupt flag clear register (LPUART_ICR) 1407
37.7.11 LPUART receive data register (LPUART_RDR) 1408
37.7.12 LPUART transmit data register (LPUART_TDR) 1408
37.7.13 LPUART prescaler register (LPUART_PRESC) 1409
37.7.14 LPUART autonomous mode control register (LPUART_AUTOCR) . . 1410
37.7.15 LPUART register map 1410

38 Serial peripheral interface (SPI) 1413

38.1 SPI introduction 1413
38.2 SPI main features 1413
38.3 SPI implementation 1414

38.4 SPI functional description 1415
38.4.1 SPI block diagram 1415
38.4.2 SPI pins and internal signals 1416
38.4.3 SPI communication general aspects 1418
38.4.4 Communications between one master and one slave 1418
38.4.5 Standard multislave communication 1420
38.4.6 Multimaster communication 1423
38.4.7 Slave select (NSS pin) management 1424
38.4.8 Ready pin (RDY) management 1428
38.4.9 Communication formats 1428
38.4.10 Configuring the SPI 1430
38.4.11 Enabling the SPI 1431
38.4.12 SPI data transmission and reception procedures 1432
38.4.13 Disabling the SPI 1436
38.4.14 Communication using DMA (direct memory addressing) 1437
38.4.15 Autonomous mode 1438
38.5 SPI specific modes and control 1440
38.5.1 TI mode 1440
38.5.2 SPI error flags 1440
38.5.3 CRC computation 1444
38.6 SPI in low-power modes 1445
38.7 SPI interrupts 1445
38.8 SPI registers 1447
38.8.1 SPI control register 1 (SPI_CR1) 1447
38.8.2 SPI control register 2 (SPI_CR2) 1449
38.8.3 SPI configuration register 1 (SPI_CFG1) 1449
38.8.4 SPI configuration register 2 (SPI_CFG2) 1451
38.8.5 SPI interrupt enable register (SPI_IER) 1453
38.8.6 SPI status register (SPI_SR) 1454
38.8.7 SPI interrupt/status flags clear register (SPI_IFCR) 1457
38.8.8 SPI autonomous mode control register (SPI_AUTOCR) 1457
38.8.9 SPI transmit data register (SPI_TXDR) 1458
38.8.10 SPI receive data register (SPI_RXDR) 1459
38.8.11 SPI polynomial register (SPI_CRCPOLY) 1459
38.8.12 SPI transmitter CRC register (SPI_TXCRC) 1460
38.8.13 SPI receiver CRC register (SPI_RXCRC) 1460

38.8.14 SPI underrun data register (SPI_UDRDR) 1461
38.8.15 SPI register map 1462

39 Serial audio interface (SAI) 1463

39.1 SAI introduction 1463
39.2 SAI main features 1463
39.3 SAI implementation 1464
39.4 SAI functional description 1465
39.4.1 SAI block diagram 1465
39.4.2 SAI pins and internal signals 1466
39.4.3 Main SAI modes 1466
39.4.4 SAI synchronization mode 1467
39.4.5 Audio data size 1468
39.4.6 Frame synchronization 1468
39.4.7 Slot configuration 1471
39.4.8 SAI clock generator 1473
39.4.9 Internal FIFOs 1476
39.4.10 PDM interface 1478
39.4.11 AC'97 link controller 1486
39.4.12 SPDIF output 1487
39.4.13 Specific features 1490
39.4.14 Error flags 1494
39.4.15 Disabling the SAI 1497
39.4.16 SAI DMA interface 1497
39.5 SAI interrupts 1498
39.6 SAI registers 1500
39.6.1 SAI configuration register 1 (SAI_ACR1) 1500
39.6.2 SAI configuration register 2 (SAI_ACR2) 1502
39.6.3 SAI frame configuration register (SAI_AFRCR) 1504
39.6.4 SAI slot register (SAI_ASLOTR) 1505
39.6.5 SAI interrupt mask register (SAI_AIM) 1506
39.6.6 SAI status register (SAI_ASR) 1508
39.6.7 SAI clear flag register (SAI_ACLRFR) 1510
39.6.8 SAI data register (SAI_ADR) 1511
39.6.9 SAI configuration register 1 (SAI_BCR1) 1511
39.6.10 SAI configuration register 2 (SAI_BCR2) 1514

39.6.11 SAI frame configuration register (SAI_BFRCR) 1516
39.6.12 SAI slot register (SAI_BSLOTR) 1517
39.6.13 SAI interrupt mask register (SAI_BIM) 1518
39.6.14 SAI status register (SAI_BSR) 1519
39.6.15 SAI clear flag register (SAI_BCLRFR) 1521
39.6.16 SAI data register (SAI_BDR) 1522
39.6.17 SAI PDM control register (SAI_PDMCR) 1523
39.6.18 SAI PDM delay register (SAI_PDMDLY) 1524
39.6.19 SAI register map 1526

40 Universal serial bus full-speed host/device interface (USB) 1528

40.1 USB introduction 1528
40.2 USB main features 1528
40.3 USB implementation 1528
40.4 USB functional description 1529
40.4.1 USB block diagram 1529
40.4.2 USB pins and internal signals 1530
40.4.3 USB reset and clocks 1530
40.4.4 General description and Device mode functionality 1530
40.4.5 Description of USB blocks used in both Device and Host modes . . . 1531
40.4.6 Description of host frame scheduler (HFS) specific to Host mode . . . 1532
40.5 Programming considerations for Device and Host modes 1533
40.5.1 Generic USB Device programming 1533
40.5.2 System and power-on reset 1534
40.5.3 Double-buffered endpoints and usage in Device mode 1541
40.5.4 Double buffered channels: usage in Host mode 1543
40.5.5 Isochronous transfers in Device mode 1544
40.5.6 Isochronous transfers in Host mode 1545
40.5.7 Suspend/resume events 1546
40.6 USB registers 1549
40.6.1 USB control register (USB_CNTR) 1550
40.6.2 USB interrupt status register (USB_ISTR) 1553
40.6.3 USB frame number register (USB_FNR) 1557
40.6.4 USB Device address (USB_DADDR) 1557
40.6.5 USB LPM control and status register (USB_LPMCSR) 1558
40.6.6 USB battery charging detector (USB_BCDR) 1559

40.6.7 USB endpoint/channel n register (USB_CHEPnR) 1560
40.7 USBSRAM registers 1569
40.7.1 Channel/endpoint transmit buffer descriptor n (USBSRAM_CHEP_TXRXBD_n) 1570
40.7.2 Channel/endpoint receive buffer descriptor n [alternate] (USBSRAM_CHEP_TXRXBD_n) 1570
40.7.3 Channel/endpoint receive buffer descriptor n (USBSRAM_CHEP_RXTXBD_n) 1572
40.7.4 Channel/endpoint transmit buffer descriptor n [alternate] (USBSRAM_CHEP_RXTXBD_n) 1573
40.7.5 USBSRAM register map 1574

41 Debug support (DBG) 1575
41.1 DBG introduction and main features 1575
41.2 DBG functional description 1576
41.2.1 DBG block diagram 1576
41.2.2 DBG pins and internal signals 1576
41.2.3 DBG reset and clocks 1577
41.2.4 DBG power domains 1577
41.2.5 DBG low-power modes 1577
41.2.6 Security 1578
41.2.7 Serial-wire and JTAG debug port 1579
41.2.8 JTAG debug port 1580
41.2.9 Serial-wire debug port 1582
41.3 Debug port registers 1583
41.3.1 DP identification register [alternate] (DP_DPIDR) 1584
41.3.2 DP abort register [alternate] (DP_ABORTR) 1585
41.3.3 DP control and status register (DP_CTRLSTATR) 1585
41.3.4 DP data link control register (DP_DLCR) 1587
41.3.5 DP target identification register (DP_TARGETIDR) 1587
41.3.6 DP data link protocol identification register (DP_DLPIDR) 1588
41.3.7 DP resend register (DP_EVENSTATR) 1588
41.3.8 DP resend register (DP_RESENDR) 1589
41.3.9 DP access port select register (DP_SELECTR) 1589
41.3.10 DP read buffer register (DP_BUFFR) 1590
41.3.11 DP register map and reset values 1590
41.4 Access port 1592
41.4.1 Access port registers 1592

41.4.2 AP control/status word register (APx_CSWR) (x = 0 to 1) 1593
41.4.3 AP transfer address register (APx_TAR) (x = 0 to 1) 1594
41.4.4 AP data read/write register (APx_DRWR) (x = 0 to 1) 1594
41.4.5 AP banked data registers y (APx_BDyR) (x = 0 to 1) 1594
41.4.6 AP configuration register (APx_CFGR) (x = 0 to 1) 1595
41.4.7 AP base address register (APx_BASER) (x = 0 to 1) 1595
41.4.8 AP identification register (APx_IDR) (x = 0 to 1) 1596
41.4.9 Access port register map and reset values 1597
41.5 System debug AP0 features 1598
41.5.1 System debug ROM table 1598
41.5.2 System debug memory type register (SYSROM_MEMTYPER) 1599
41.5.3 Sytem debug CoreSight peripheral identity register 4 (SYSROM_PIDR4) 1599
41.5.4 System debug CoreSight peripheral identity register 0 (SYSROM_PIDR0) 1600
41.5.5 System debug CoreSight peripheral identity register 1 (SYSROM_PIDR1) 1600
41.5.6 System debug CoreSight peripheral identity register 2 (SYSROM_PIDR2) 1601
41.5.7 System debug CoreSight peripheral identity register 3 (SYSROM_PIDR3) 1601
41.5.8 System debug CoreSight component identity register 0 (SYSROM_CIDR0) 1602
41.5.9 System debug CoreSight peripheral identity register 1 (SYSROM_CIDR1) 1602
41.5.10 System debug CoreSight component identity register 2 (SYSROM_CIDR2) 1603
41.5.11 System debug CoreSight component identity register 3 (SYSROM_CIDR3) 1603
41.5.12 System debug ROM table register map and reset values 1604
41.6 Cortex-M33 AP1 features 1605
41.6.1 CPU ROM tables 1605
41.6.2 MCU and processor ROM memory type register (ROM_MEMTYPER) 1608
41.6.3 MCU and processor ROM CoreSight peripheral identity register 4 (ROM_PIDR4) 1608
41.6.4 MCU and processor ROM CoreSight peripheral identity register 0 (ROM_PIDR0) 1609
41.6.5 MCU and processor ROM CoreSight peripheral identity register 1 (ROM_PIDR1) 1609

41.6.6 MCU and processor ROM CoreSight peripheral identity register 2 (ROM_PIDR2) 1610
41.6.7 MCU and processor ROM CoreSight peripheral identity register 3 (ROM_PIDR3) 1610
41.6.8 MCU and processor ROM CoreSight component identity register 0 (ROM_CIDR0) 1611
41.6.9 MCU and processor ROM CoreSight peripheral identity register 1 (ROM_CIDR1) 1611
41.6.10 MCU and processor ROM CoreSight component identity register 2 (ROM_CIDR2) 1612
41.6.11 MCU and processor ROM CoreSight component identity register 3 (ROM_CIDR3) 1612
41.6.12 MCU and processor ROM tables register map and reset values 1613
41.7 Data watchpoint and trace unit (DWT) 1614
41.7.1 DWT control register (DWT_CTRLR) 1615
41.7.2 DWT cycle count register (DWT_CYCCNTR) 1616
41.7.3 DWT CPI count register (DWT_CPICNTR) 1617
41.7.4 DWT exception count register (DWT_EXCCNTR) 1617
41.7.5 DWT sleep count register (DWT_SLPCNTR) 1617
41.7.6 DWT LSU count register (DWT_LSUCNTR) 1618
41.7.7 DWT fold count register (DWT_FOLDCNTR) 1618
41.7.8 DWT program counter sample register (DWT_PCSR) 1619
41.7.9 DWT comparator register x (DWT_COMPxR) 1619
41.7.10 DWT function register 0(DWT_FUNCTR0) 1619
41.7.11 DWT device type architecture register (DWT_DEVARCHR) 1623
41.7.12 DWT device type register 4 (DWT_DEVTYPER) 1624
41.7.13 DWT CoreSight peripheral identity register 4 (DWT_PIDR4) 1624
41.7.14 DWT CoreSight peripheral identity register 0 (DWT_PIDR0) 1625
41.7.15 DWT CoreSight peripheral identity register 1 (DWT_PIDR1) 1625
41.7.16 DWT CoreSight peripheral identity register 2 (DWT_PIDR2) 1626
41.7.17 DWT CoreSight peripheral identity register 3 (DWT_PIDR3) 1626
41.7.18 DWT CoreSight component identity register 0 (DWT_CIDR0) 1627
41.7.19 DWT CoreSight peripheral identity register 1 (DWT_CIDR1) 1627
41.7.20 DWT CoreSight component identity register 2 (DWT_CIDR2) 1627
41.7.21 DWT CoreSight component identity register 3 (DWT_CIDR3) 1628
41.7.22 DWT register map and reset values 1629
41.8 Instrumentation trace macrocell (ITM) 1632
41.8.1 ITM registers 1632
41.8.2 ITM stimulus register x (ITM_STIMRx) 1632

41.8.3 ITM trace enable register (ITM_TER) 1633
41.8.4 ITM trace privilege register (ITM_TPR) 1633
41.8.5 ITM trace control register (ITM_TCR) 1634
41.8.6 ITM device type architecture register (ITM_DEVARCHR) 1635
41.8.7 ITM device type register 4 (ITM_DEVTYPER) 1635
41.8.8 ITM CoreSight peripheral identity register 4 (ITM_PIDR4) 1636
41.8.9 ITM CoreSight peripheral identity register 0 (ITM_PIDR0) 1636
41.8.10 ITM CoreSight peripheral identity register 1 (ITM_PIDR1) 1636
41.8.11 ITM CoreSight peripheral identity register 2 (ITM_PIDR2) 1637
41.8.12 ITM CoreSight peripheral identity register 3 (ITM_PIDR3) 1637
41.8.13 ITM CoreSight component identity register 0 (ITM_CIDR0) 1638
41.8.14 ITM CoreSight peripheral identity register 1 (ITM_CIDR1) 1638
41.8.15 ITM CoreSight component identity register 2 (ITM_CIDR2) 1638
41.8.16 ITM CoreSight component identity register 3 (ITM_CIDR3) 1639
41.8.17 ITM register map and reset values 1640
41.9 Breakpoint unit (BPU) 1642
41.9.1 BPU control register (BPU_CTRLR) 1642
41.9.2 BPU comparator x register (BPU_COMPxR) 1642
41.9.3 BPU device type architecture register (BPU_DEVARCHR) 1643
41.9.4 BPU device type register 4 (BPU_DEVTYPER) 1643
41.9.5 BPU CoreSight peripheral identity register 4 (BPU_PIDR4) 1644
41.9.6 BPU CoreSight peripheral identity register 0 (BPU_PIDR0) 1644
41.9.7 BPU CoreSight peripheral identity register 1 (BPU_PIDR1) 1644
41.9.8 BPU CoreSight peripheral identity register 2 (BPU_PIDR2) 1645
41.9.9 BPU CoreSight peripheral identity register 3 (BPU_PIDR3) 1645
41.9.10 BPU CoreSight component identity register 0 (BPU_CIDR0) 1646
41.9.11 BPU CoreSight peripheral identity register 1 (BPU_CIDR1) 1646
41.9.12 BPU CoreSight component identity register 2 (BPU_CIDR2) 1646
41.9.13 BPU CoreSight component identity register 3 (BPU_CIDR3) 1647
41.9.14 BPU register map and reset values 1648
41.10 Embedded trace macrocell (ETM) 1649
41.10.1 ETM registers 1649
41.10.2 ETM register map and reset values 1672
41.11 Trace port interface unit (TPIU) 1675
41.11.1 TPIU registers 1676
41.11.2 TPIU supported port size register (TPIU_SSPSR) 1676
41.11.3 TPIU current port size register (TPIU_CSPSR) 1677

41.11.4 TPIU asynchronous clock prescaler register (TPIU_ACPR) 1677
41.11.5 TPIU selected pin protocol register (TPIU_SPPR) 1677
41.11.6 TPIU formatter and flush status register (TPIU_FFSR) 1678
41.11.7 TPIU formatter and flush control register (TPIU_FFCR) 1678
41.11.8 TPIU formatter synchronization counter register (TPIU_FSCR) 1679
41.11.9 TPIU claim tag set register (TPIU_CLAIMSETR) 1680
41.11.10 TPIU claim tag clear register (TPIU_CLAIMCLR) 1680
41.11.11 TPIU device configuration register (TPIU_DEVIDR) 1681
41.11.12 TPIU device type identifier register (TPIU_DEVTYPER) 1681
41.11.13 TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4) 1682
41.11.14 TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0) 1682
41.11.15 TPIU CoreSight peripheral identity register 1 (TPIU_PIDR1) 1682
41.11.16 TPIU CoreSight peripheral identity register 2 (TPIU_PIDR2) 1683
41.11.17 TPIU CoreSight peripheral identity register 3 (TPIU_PIDR3) 1683
41.11.18 TPIU CoreSight component identity register 0 (TPIU_CIDR0) 1684
41.11.19 TPIU CoreSight peripheral identity register 1 (TPIU_CIDR1) 1684
41.11.20 TPIU CoreSight component identity register 2 (TPIU_CIDR2) 1684
41.11.21 TPIU CoreSight component identity register 3 (TPIU_CIDR3) 1685
41.11.22 TPIU register map and reset values 1686
41.12 Cross trigger interface (CTI) 1688
41.12.1 CTI registers 1689
41.13 Microcontroller debug unit (DBGMCU) 1702
41.13.1 DBGMCU access 1702
41.13.2 Device ID 1702
41.13.3 Part number codification 1702
41.13.4 Low-power mode emulation 1702
41.13.5 Low-power mode status 1704
41.13.6 Peripheral clock freeze 1704
41.13.7 DBGMCU registers 1705
41.13.8 DBGMCU register map and reset values 1717
41.14 References 1719

42 Device electronic signature (DESIG) 1720

42.1 Device electronic signature registers 1720
42.1.1 DESIG package data register (DESIG_PKGR) 1720
42.1.2 DESIG 96-bit unique device ID register 1 (DESIG_UIDR1) 1720
42.1.3 DESIG 96-bit unique device ID register 2 (DESIG_UIDR2) 1721

42.1.4 DESIG 96-bit unique device ID register 3 (DESIG_UIDR3) 1721
42.1.5 DESIG temperature calibration 1 register (DESIG_TSCAL1R) 1722
42.1.6 DESIG temperature calibration 2 register (DESIG_TSCAL2R) 1722
42.1.7 DESIG flash size data register (DESIG_FLASHSIZER) 1722
42.1.8 DESIG internal voltage reference calibration register (DESIG_VREFINTCALR) 1723
42.1.9 DESIG IEEE 64-bit unique device ID register 1 (DESIG_UID64R1) . 1723
42.1.10 DESIG IEEE 64-bit unique device ID register 2 (DESIG_UID64R2) . 1723
42.1.11 DESIG register map 1724

43 Revision history 1725

List of tables

Table 1. Bus matrix acces arbitration . . . 70
Table 2. Memory map security attribution example vs. SAU configuration regions . . . 71
Table 3. Securable peripherals by TZSC . . . 72
Table 4. TrustZone®-aware peripherals . . . 73
Table 5. Memory map and peripheral register boundary addresses . . . 77
Table 6. Boot modes when TrustZone® is disabled (TZEN = 0) . . . 83
Table 7. Boot modes when TrustZone® is enabled (TZEN = 1) . . . 83
Table 8. Boot space versus RDP protection . . . 84
Table 9. Configuring security attributes with IDAU and SAU . . . 90
Table 10. MPCWM resources . . . 92
Table 11. MPCBB resources . . . 92
Table 12. DMA channel use (security) . . . 95
Table 13. Secure alternate function between peripherals and allocated I/Os . . . 98
Table 14. Nonsecure peripheral functions not connected to secure I/Os . . . 98
Table 15. Nonsecure peripheral functions that can be connected to secure I/Os . . . 99
Table 16. TrustZone-aware DBGMCU accesses management . . . 100
Table 17. DMA channel use (privilege) . . . 105
Table 18. Internal tampers in TAMP . . . 108
Table 19. Accelerated cryptographic operations . . . 111
Table 20. Main product life-cycle transitions . . . 113
Table 21. Typical product life-cycle phases . . . 113
Table 22. OEM key RDP unlocking methods . . . 115
Table 23. Debug protection with RDP . . . 117
Table 24. Software intellectual property protection with RDP . . . 118
Table 25. GTZC features . . . 122
Table 26. GTZC sub-blocks address offset . . . 123
Table 27. MPCWM resource assignment . . . 123
Table 28. MPCBB resource assignment . . . 123
Table 29. Secure properties of sub-regions A and B . . . 126
Table 30. Privileged properties of sub-regions A and B . . . 126
Table 31. GTZC interrupt request . . . 128
Table 32. GTZC1 TZSC register map and reset values . . . 140
Table 33. GTZC1 TZIC register map and reset values . . . 157
Table 34. GTZC1 MPCBB1 register map and reset values . . . 162
Table 35. GTZC1 MPCBB2 register map and reset values . . . 163
Table 36. Internal SRAMs features . . . 165
Table 37. SRAM access parity errors . . . 166
Table 38. SRAM parity error bus master ID . . . 166
Table 39. Number of wait states versus hclk frequency and voltage range scaling . . . 166
Table 40. Effect of low-power modes on RAMCFG . . . 167
Table 41. RAMCFG interrupt requests . . . 167
Table 42. RAMCFG register map and reset values . . . 174
Table 43. Flash module 1-Mbyte single bank organization . . . 176
Table 44. Number of wait states according to CPU clock (hclk1) frequency (LPM = 0) . . . 177
Table 45. Number of wait states according to CPU clock (hclk1) frequency (LPM = 1) . . . 178
Table 46. Program and erase suspend control . . . 187
Table 47. Flash operation interrupted by a system reset . . . 190
Table 48. User option byte organization mapping . . . 191

Table 49. Default secure option bytes after TZEN activation . . . 194
Table 50. Secure watermark-based area . . . 195
Table 51. Secure hide protection . . . 196
Table 52. Write access to HDP_PEXT and HDPEXT_ACCDIS . . . 197
Table 53. Secure hide protection extension . . . 198
Table 54. Flash security state . . . 199
Table 55. WRP protection . . . 201
Table 56. Flash memory readout protection status (TZEN=0) . . . 202
Table 57. Access status versus protection level and execution modes when TZEN = 0 . . . 203
Table 58. Flash memory readout protection status (TZEN = 1) . . . 204
Table 59. Access status versus protection level and execution modes when TZEN = 1 . . . 205
Table 60. Flash memory access versus RDP level when TrustZone is active (TZEN = 1) . . . 211
Table 61. Flash memory secure access versus HDP and HDP extension (TZEN = 1) . . . 212
Table 62. Flash memory access versus RDP level when TrustZone is disabled (TZEN = 0) . . . 212
Table 63. Flash memory mass erase versus RDP level when TrustZone is active (TZEN = 1) . . . 213
Table 64. Flash system memory, OTP and RSS accesses . . . 213
Table 65. Flash page access versus privilege mode . . . 214
Table 66. Flash mass erase versus privilege mode . . . 214
Table 67. Flash registers access . . . 214
Table 68. SECBBRx registers access when TrustZone is active (TZEN = 1) . . . 214
Table 69. PRIVBBRx registers access when TrustZone is active (TZEN = 1) . . . 215
Table 70. PRIVBBRx registers access when TrustZone is disabled (TZEN = 0) . . . 215
Table 71. Flash interrupt requests . . . 215
Table 72. FLASH register map and reset values . . . 245
Table 73. ICACHE features . . . 250
Table 74. TAG memory dimensioning parameters for n-way set associative operating mode (default) . . . 252
Table 75. TAG memory dimensioning parameters for direct-mapped cache mode . . . 253
Table 76. ICACHE cacheability for AHB transaction . . . 255
Table 77. Memory configurations . . . 255
Table 78. ICACHE remap region size, base address, and remap address . . . 256
Table 79. ICACHE interrupts . . . 260
Table 80. ICACHE register map and reset values . . . 264
Table 81. Radio features . . . 267
Table 82. Input/output pins . . . 267
Table 83. PA output power table format . . . 268
Table 84. 2.4 GHz RADIO supply configuration . . . 269
Table 85. Effect of low-power modes on the 2.4 GHz RADIO . . . 270
Table 86. 2.4 PTACONV input/output pins . . . 272
Table 87. PTACONV internal input/output signals . . . 272
Table 88. 2.4 PTACONV timing parameters . . . 276
Table 89. Effect of low-power modes on the PTACONV . . . 277
Table 90. PTACONV register map and reset values . . . 281
Table 91. PWR input/output pins . . . 283
Table 92. PWR internal input/output signals . . . 283
Table 93. PWR wake-up source selection . . . 284
Table 94. Low-power mode summary . . . 298
Table 95. Functionalities depending on the working mode . . . 299
Table 96. Sleep mode . . . 306
Table 97. Stop 0 mode . . . 309
Table 98. Stop 1 mode . . . 311
Table 99. Stop 2 mode . . . 313

Table 100. Stop 3 mode . . . 315
Table 101. GPIO retention pin with pull-up and pull-down . . . 317
Table 102. Standby mode . . . 319
Table 103. Power modes output states versus MCU power modes . . . 320
Table 104. PWR Security configuration summary . . . 321
Table 105. PWR interrupt requests . . . 322
Table 106. PWR register map and reset values . . . 347
Table 107. RCC input/output signals connected to package pins or balls . . . 350
Table 108. LSI clock selection . . . 361
Table 109. SYSCLK and bus maximum frequency . . . 362
Table 110. PLL1RCLKPRERDY delay versus pll1rclk frequency . . . 363
Table 111. PLL1RCLKPRERDY and SYSCLK behavior . . . 363
Table 112. Clock source maximum frequency . . . 365
Table 113. 2.4 GHz RADIO bus clock control . . . 367
Table 114. Autonomous peripherals . . . 375
Table 115. RCC security configuration summary . . . 377
Table 116. Interrupt sources and control . . . 380
Table 117. RCC register map and reset values . . . 451
Table 118. GPIO implementation . . . 458
Table 119. Port bit configuration . . . 460
Table 120. GPIO secured bits . . . 468
Table 121. GPIOA register map and reset values . . . 500
Table 122. GPIOB register map and reset values . . . 501
Table 123. GPIOC register map and reset values . . . 502
Table 124. GPIOH register map and reset values . . . 503
Table 125. Effect of low-power modes on I/O compensation . . . 505
Table 126. TrustZone security and privilege register access . . . 505
Table 127. BOOSTEN and ANASWVDD set/cleared . . . 509
Table 128. SYSCFG register map and reset values . . . 520
Table 129. Peripherals interconnect matrix . . . 521
Table 130. LPDMA1 channels implementation . . . 529
Table 131. LPDMA1 autonomous mode and wake-up in low-power modes . . . 529
Table 132. Programmed LPDMA1 request . . . 530
Table 133. Programmed LPDMA1 request as a block request . . . 531
Table 134. Programmed LPDMA1 trigger . . . 532
Table 135. Programmed LPDMA source/destination single . . . 551
Table 136. Programmed data handling . . . 552
Table 137. Effect of low-power modes on LPDMA . . . 564
Table 138. LPDMA interrupt requests . . . 565
Table 139. LPDMA register map and reset values . . . 584
Table 140. Vector table . . . 586
Table 141. EXTI pin overview . . . 591
Table 142. EVG pin overview . . . 591
Table 143. EXTI line connections . . . 592
Table 144. Masking functionality . . . 594
Table 145. Register protection overview . . . 595
Table 146. EXTI register map sections . . . 597
Table 147. EXTI register map and reset values . . . 609
Table 148. CRC internal input/output signals . . . 612
Table 149. CRC register map and reset values . . . 617
Table 150. XSPI implementation . . . 618
Table 151. XSPI input/output pins . . . 620

Table 152. XSPI internal signals . . . 620
Table 153. OctaRAM command address bit assignment (based on 64 Mb OctaRAM) . . . 633
Table 154. Address alignment cases . . . 637
Table 155. XSPI interrupt requests . . . 638
Table 156. XSPI register map and reset values . . . 651
Table 157. STM32WBA2 features . . . 653
Table 158. DLYB internal input/output signals . . . 654
Table 159. DLYB interconnection . . . 654
Table 160. Delay block control . . . 655
Table 161. DLYB register map and reset values . . . 658
Table 162. ADC features . . . 660
Table 163. Memory location of the temperature sensor calibration values . . . 661
Table 164. Memory location of the internal reference voltage sensor calibration value . . . 661
Table 165. ADC input/output pins . . . 662
Table 166. ADC internal input/output signals . . . 663
Table 167. ADC interconnection . . . 663
Table 168. Latency between trigger and start of conversion . . . 668
Table 169. Configuring the trigger polarity . . . 675
Table 170. t_{SAR} timings depending on resolution . . . 677
Table 171. Analog watchdog comparison . . . 689
Table 172. Analog watchdog 1 channel selection . . . 689
Table 173. Maximum output results vs N and M. Grayed values indicates truncation . . . 693
Table 174. Effect of low-power modes on the ADC . . . 698
Table 175. ADC wake-up and interrupt requests . . . 699
Table 176. ADC register map and reset values . . . 719
Table 177. RNG internal input/output signals . . . 723
Table 178. RNG interrupt requests . . . 732
Table 179. RNG configurations . . . 733
Table 180. Additional health test configurations . . . 733
Table 181. Configuration selection . . . 733
Table 182. RNG register map and reset map . . . 741
Table 183. AES features . . . 743
Table 184. AES internal input/output signals . . . 743
Table 185. AES approved symmetric key functions . . . 744
Table 186. Counter mode initialization vector definition . . . 753
Table 187. Initialization of IV registers in GCM mode . . . 756
Table 188. GCM last block definition . . . 756
Table 189. Initialization of IV registers in CCM mode . . . 763
Table 190. AES data swapping example . . . 765
Table 191. Key endianness in AES_KEYRx registers (128/256-bit keys) . . . 768
Table 192. IVI bitfield spread over AES_IVRx registers . . . 768
Table 193. Effect of low-power modes on AES . . . 769
Table 194. AES interrupt requests . . . 770
Table 195. Processing latency for ECB, CBC and CTR . . . 771
Table 196. Processing latency for GCM and CCM (in clock cycles) . . . 771
Table 197. AES register map and reset values . . . 783
Table 198. HASH internal input/output signals . . . 786
Table 199. Information on supported hash algorithms . . . 787
Table 200. Hash processor outputs . . . 790
Table 201. Processing time (in clock cycle) . . . 795

Table 202. HASH interrupt requests . . . 796
Table 203. HASH1 register map and reset values . . . 804
Table 204. OTFDEC internal input/output signals . . . 807
Table 205. OTFDEC interrupt requests . . . 811
Table 206. OTFDEC register map and reset values . . . 825
Table 207. Internal input/output signals . . . 830
Table 208. PKA integer arithmetic functions list . . . 831
Table 209. PKA prime field (Fp) elliptic curve functions list . . . 832
Table 210. Example of ‘a’ curve coefficient for ECC Fp scalar . . . 838
Table 211. Montgomery parameter computation . . . 838
Table 212. Modular addition . . . 839
Table 213. Modular subtraction . . . 839
Table 214. Montgomery multiplication . . . 840
Table 215. Modular exponentiation (normal mode) . . . 841
Table 216. Modular exponentiation (fast mode) . . . 841
Table 217. Modular exponentiation (protected mode) . . . 842
Table 218. Modular inversion . . . 842
Table 219. Modular reduction . . . 843
Table 220. Arithmetic addition . . . 843
Table 221. Arithmetic subtraction . . . 843
Table 222. Arithmetic multiplication . . . 844
Table 223. Arithmetic comparison . . . 844
Table 224. CRT exponentiation . . . 845
Table 225. Point on elliptic curve Fp check . . . 846
Table 226. ECC Fp scalar multiplication . . . 846
Table 227. ECDSA sign - Inputs . . . 848
Table 228. ECDSA sign - Outputs . . . 848
Table 229. Extended ECDSA sign - additional outputs . . . 849
Table 230. ECDSA verification - inputs . . . 849
Table 231. ECDSA verification - outputs . . . 850
Table 232. ECC complete addition . . . 850
Table 233. ECC double base ladder . . . 851
Table 234. ECC projective to affine . . . 852
Table 235. Family of supported curves for ECC operations . . . 853
Table 236. Modular exponentiation . . . 854
Table 237. ECC scalar multiplication . . . 854
Table 238. ECDSA signature average computation time . . . 855
Table 239. ECDSA verification average computation times . . . 855
Table 240. ECC double base ladder average computation times . . . 855
Table 241. ECC projective to affine average computation times . . . 855
Table 242. ECC complete addition average computation times . . . 855
Table 243. Point on elliptic curve Fp check average computation times . . . 855
Table 244. Montgomery parameters average computation times . . . 856
Table 245. Effect of low-power modes on PKA . . . 856
Table 246. PKA interrupt requests . . . 856
Table 247. PKA register map and reset values . . . 861
Table 248. General purpose timers . . . 863
Table 249. TIM input/output pins . . . 865
Table 250. TIM internal input/output signals . . . 865
Table 251. Interconnect to the tim_ti1 input multiplexer . . . 866
Table 252. Interconnect to the tim_ti2 input multiplexer . . . 866
Table 253. Interconnect to the tim_ti3 input multiplexer . . . 866

Table 254. Interconnect to the tim_ti4 input multiplexer . . . 866
Table 255. TIMx internal trigger connection . . . 867
Table 256. Interconnect to the tim_etr input multiplexer . . . 867
Table 257. Interconnect to the tim_ocref_clr input multiplexer . . . 867
Table 258. CCR and ARR register change dithering pattern . . . 898
Table 259. CCR register change dithering pattern in center-aligned PWM mode . . . 899
Table 260. Counting direction versus encoder signals(CC1P = CC2P = 0) . . . 908
Table 261. Counting direction versus encoder signals and polarity settings . . . 913
Table 262. DMA request . . . 938
Table 263. Effect of low-power modes on TIM2 . . . 938
Table 264. Interrupt requests . . . 939
Table 265. Output control bit for standard tim_ocx channels . . . 960
Table 266. TIM2 register map and reset values . . . 972
Table 267. TIM16/TIM17 . . . 975
Table 268. TIM input/output pins . . . 976
Table 269. TIM internal input/output signals . . . 977
Table 270. Interconnect to the tim_ti1 input multiplexer . . . 978
Table 271. Timer break interconnect . . . 978
Table 272. System break interconnect . . . 978
Table 273. Interconnect to the ocref_clr input multiplexer . . . 978
Table 274. CCR and ARR register change dithering pattern . . . 997
Table 275. Break protection disarming conditions . . . 1005
Table 276. DMA request . . . 1012
Table 277. Effect of low-power modes on TIM16/TIM17 . . . 1012
Table 278. Interrupt requests . . . 1013
Table 279. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) . . . 1024
Table 280. TIM16/TIM17 register map and reset values . . . 1038
Table 281. LPTIM device implementation . . . 1041
Table 282. LPTIM features . . . 1041
Table 283. LPTIM1/2 input/output pins . . . 1042
Table 284. LPTIM1/2 internal signals . . . 1043
Table 285. LPTIM1/2 external trigger connections . . . 1044
Table 286. LPTIM1/2 input 1 connections . . . 1044
Table 287. LPTIM1/2 input 2 connections . . . 1044
Table 288. LPTIM1/2 input capture 1 connections . . . 1044
Table 289. LPTIM1/2 input capture 2 connections . . . 1044
Table 290. Prescaler division ratios . . . 1046
Table 291. Encoder counting scenarios . . . 1053
Table 292. Input capture Glitch filter latency (in counter step unit) . . . 1057
Table 293. Effect of low-power modes on the LPTIM . . . 1062
Table 294. Interrupt events . . . 1063
Table 295. LPTIM register map and reset values . . . 1084
Table 296. IWDG features . . . 1087
Table 297. IWDG delays versus actions . . . 1088
Table 298. IWDG internal input/output signals . . . 1089
Table 299. Effect of low power modes on IWDG . . . 1093
Table 300. IWDG interrupt request . . . 1095
Table 301. IWDG register map and reset values . . . 1101
Table 302. RTC input/output pins . . . 1104
Table 303. RTC internal input/output signals . . . 1104
Table 304. RTC interconnection . . . 1105

Table 305. RTC pin PC13 configuration . . . 1105
Table 306. RTC_OUT mapping . . . 1107
Table 307. Effect of low-power modes on RTC . . . 1123
Table 308. RTC pins functionality over modes . . . 1123
Table 309. Nonsecure interrupt requests . . . 1124
Table 310. Secure interrupt requests . . . 1124
Table 311. RTC register map and reset values . . . 1154
Table 312. TAMP input/output pins . . . 1158
Table 313. TAMP internal input/output signals . . . 1158
Table 314. TAMP interconnection . . . 1159
Table 315. Device resource x tamper protection . . . 1164
Table 316. Active tamper output change period . . . 1168
Table 317. Minimum ATPER value . . . 1169
Table 318. Active tamper filtered pulse duration . . . 1170
Table 319. Effect of low-power modes on TAMP . . . 1171
Table 320. TAMP pins functionality over modes . . . 1171
Table 321. Interrupt requests . . . 1172
Table 322. TAMP register map and reset values . . . 1198
Table 323. I2C implementation . . . 1201
Table 324. I2C input/output pins . . . 1202
Table 325. I2C internal input/output signals . . . 1203
Table 326. I2C1 interconnection . . . 1203
Table 327. I2C3 interconnection . . . 1203
Table 328. Comparison of analog and digital filters . . . 1206
Table 329. I²C-bus and SMBus specification data setup and hold times . . . 1208
Table 330. I2C configuration . . . 1212
Table 331. I²C-bus and SMBus specification clock timings . . . 1223
Table 332. Timing settings for f_{I2CCLK} of 8 MHz . . . 1233
Table 333. Timing settings for f_{I2CCLK} of 16 MHz . . . 1233
Table 334. SMBus timeout specifications . . . 1235
Table 335. SMBus with PEC configuration . . . 1237
Table 336. TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms . . . 1238
Table 337. TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms . . . 1238
Table 338. TIMEOUTA[11:0] for maximum t_{IDLE} of 50 µs . . . 1238
Table 339. Effect of low-power modes to I2C . . . 1248
Table 340. I2C interrupt requests . . . 1249
Table 341. I2C register map and reset values . . . 1266
Table 342. Instance implementation on STM32WBA2 . . . 1269
Table 343. USART/LPUART features . . . 1269
Table 344. USART/UART input/output pins . . . 1272
Table 345. USART internal input/output signals . . . 1273
Table 346. USART interconnection (USART1) . . . 1273
Table 347. Noise detection from sampled data . . . 1285
Table 348. Tolerance of the USART receiver when BRR [3:0] = 0000 . . . 1289
Table 349. Tolerance of the USART receiver when BRR[3:0] is different from 0000 . . . 1289
Table 350. USART frame formats . . . 1294
Table 351. Effect of low-power modes on the USART . . . 1315
Table 352. USART interrupt requests . . . 1316
Table 353. USART register map and reset values . . . 1355
Table 354. Instance implementation on STM32WBA2 . . . 1358
Table 355. USART/LPUART features . . . 1358
Table 356. LPUART input/output pins . . . 1360

Table 357. LPUART internal input/output signals . . . 1360
Table 358. LPUART interconnections (LPUART1) . . . 1361
Table 359. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz . . . 1372
Table 360. Tolerance of the LPUART receiver . . . 1373
Table 362. Effect of low-power modes on the LPUART . . . 1384
Table 363. LPUART interrupt requests . . . 1385
Table 364. LPUART register map and reset values . . . 1410
Table 365. SPI features . . . 1414
Table 366. SPI input/output pins . . . 1416
Table 367. SPI internal input/output signals . . . 1417
Table 368. SPI interconnection (SPI3) . . . 1417
Table 369. Effect of low-power modes on the SPI . . . 1445
Table 370. SPI wake-up and interrupt requests . . . 1446
Table 371. SPI register map and reset values . . . 1462
Table 372. SAI features . . . 1464
Table 373. SAI internal input/output signals . . . 1466
Table 374. SAI input/output pins . . . 1466
Table 375. MCLK_x activation conditions . . . 1473
Table 376. Clock generator programming examples . . . 1476
Table 377. SAI_A configuration for TDM mode . . . 1483
Table 378. TDM frame configuration examples . . . 1485
Table 379. SOPD pattern . . . 1488
Table 380. Parity bit calculation . . . 1488
Table 381. Audio sampling frequency versus symbol rates . . . 1489
Table 382. SAI interrupt sources . . . 1498
Table 383. SAI register map and reset values . . . 1526
Table 384. STM32WBA2 USB implementation . . . 1528
Table 385. USB input/output pins . . . 1530
Table 386. Double-buffering buffer flag definition . . . 1542
Table 387. Bulk double-buffering memory buffers usage (Device mode) . . . 1542
Table 388. Bulk double-buffering memory buffers usage (Host mode) . . . 1544
Table 389. Isochronous memory buffers usage . . . 1545
Table 390. Isochronous memory buffers usage . . . 1546
Table 391. Resume event detection . . . 1548
Table 392. Resume event detection for host . . . 1549
Table 393. Reception status encoding . . . 1567
Table 394. Endpoint/channel type encoding . . . 1567
Table 395. Endpoint/channel kind meaning . . . 1567
Table 396. Transmission status encoding . . . 1567
Table 397. USB register map and reset values . . . 1568
Table 398. Definition of allocated buffer memory . . . 1571
Table 399. USBSRAM register map and reset values . . . 1574
Table 400. JTAG/serial-wire debug port pins . . . 1576
Table 401. Single-wire trace port pins . . . 1576
Table 402. Embedded trace port pins . . . 1577
Table 403. Authentication signal states . . . 1579
Table 404. JTAG-DP data registers . . . 1581
Table 405. Packet request . . . 1583
Table 406. ACK response . . . 1583
Table 407. Data transfer . . . 1583
Table 408. DP register map and reset values . . . 1590
Table 409. AP register map and reset values . . . 1597

Table 410. System debug ROM table . . . 1598
Table 411. MCU ROM table . . . 1605
Table 412. Processor ROM table . . . 1606
Table 413. ROM table register map and reset values . . . 1613
Table 414. DWT register map and reset values . . . 1629
Table 415. ITM register map and reset values . . . 1640
Table 416. BPU register map and reset values . . . 1648
Table 417. ETM register map and reset values . . . 1672
Table 418. TPIU register map and reset values . . . 1686
Table 419. CTI inputs . . . 1688
Table 420. CTI outputs . . . 1688
Table 421. CTI register map and reset values . . . 1700
Table 422. Low power debug overview . . . 1703
Table 423. Low-power mode status flags . . . 1704
Table 424. Peripheral clock freeze control . . . 1704
Table 425. Peripheral behavior in debug mode . . . 1705
Table 426. Debugger access to freeze register bits . . . 1705
Table 427. DBGMCU register map and reset values . . . 1717
Table 428. DESIG register map and reset values . . . 1724
Table 429. Document revision history . . . 1725

List of figures

Figure 1. System architecture . . . 69
Figure 2. Memory map . . . 76
Figure 3. Secure/nonsecure partitioning using TrustZone technology . . . 87
Figure 4. Sharing memory map between CPU in secure and nonsecure state . . . 89
Figure 5. Secure world transition and memory partitioning . . . 90
Figure 6. Global TrustZone framework and TrustZone awareness . . . 91
Figure 7. Flash memory TrustZone protections . . . 94
Figure 8. Flash memory secure HDP area . . . 102
Figure 9. Key management principle . . . 110
Figure 10. Device life-cycle security . . . 112
Figure 11. RDP level transition scheme . . . 114
Figure 12. Collaborative development principle . . . 117
Figure 13. GTZC in Armv8-M subsystem block diagram . . . 122
Figure 14. GTZC block diagram . . . 124
Figure 15. Watermark memory protection controller (region x/sub-regions A and B) . . . 126
Figure 16. MPCBB block diagram . . . 127
Figure 17. Example secure, HDP, HDP extension areas . . . 198
Figure 18. RDP level transition scheme when TrustZone is disabled (TZEN = 0) . . . 207
Figure 19. RDP level transition scheme when TrustZone is enabled (TZEN = 1) . . . 208
Figure 20. ICACHE block diagram . . . 251
Figure 21. ICACHE TAG and data memories functional view . . . 253
Figure 22. ICACHE remapping address mechanism . . . 256
Figure 23. Radio system block diagram . . . 267
Figure 24. Transmit path and output power control . . . 268
Figure 25. Bluetooth AoA/AoD antennas control . . . 269
Figure 26. PTACONV block diagram . . . 272
Figure 27. 4-wire PTA grant protocol . . . 273
Figure 28. 4-wire PTA deny protocol . . . 274
Figure 29. 3-wire time-multiplexed PTA_STATUS . . . 275
Figure 30. Power supply overview . . . 285
Figure 31. Application power supply schemes . . . 287
Figure 32. Brownout reset waveform . . . 292
Figure 33. PVD thresholds . . . 293
Figure 34. Operating modes . . . 297
Figure 35. Simplified diagram of the reset circuit . . . 351
Figure 36. Clock tree . . . 354
Figure 37. HSE32 hardware configurations . . . 356
Figure 38. LSE 32 clock sources . . . 360
Figure 39. Radio control . . . 368
Figure 40. Audi synchronization counter block diagram . . . 372
Figure 41. Audio synchronization timing example . . . 373
Figure 42. Structure of 3 or 5 V-tolerant GPIO (TT or FT) . . . 459
Figure 43. Input floating / pull-up / pull-down configurations . . . 464
Figure 44. Output configuration . . . 465
Figure 45. Alternate function configuration . . . 465
Figure 46. High-impedance analog configuration . . . 466
Figure 47. I/O compensation cell block diagram . . . 504
Figure 48. LPDMA block diagram . . . 533

Figure 49. LPDMA channel direct programming without linked-list (LPDMA_CxLLR = 0) 534
Figure 50. LPDMA channel suspend and resume sequence 535
Figure 51. LPDMA channel abort and restart sequence 536
Figure 52. Static linked-list data structure (all Uxx = 1) of channel x 537
Figure 53. LPDMA dynamic linked-list data structure of an addressing channel x 538
Figure 54. LPDMA channel execution and linked-list programming in run-to-completion mode (LPDMA_CxCR.LSM = 0) 540
Figure 55. Inserting a LLIn with an auxiliary LPDMA channel y 542
Figure 56. LPDMA channel execution and linked-list programming in link step mode (LPDMA_CxCR.LSM = 1) 544
Figure 57. Building LLIn+1: LPDMA dynamic linked-lists in link step mode 545
Figure 58. Replace with a new LLIn' in register file in link step mode 546
Figure 59. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 1) 547
Figure 60. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 2) 548
Figure 61. LPDMA channel execution and linked-list programming 550
Figure 62. LPDMA arbitration policy 554
Figure 63. Trigger hit, memorization and overrun waveform 558
Figure 64. LPDMA circular buffer programming: update of the memory start address 559
Figure 65. Shared LPDMA channel with circular buffering: update of the memory start address 560
Figure 66. EXTI block diagram 591
Figure 67. Configurable event trigger logic CPU wake-up 593
Figure 68. EXTI mux GPIO selection 594
Figure 69. CRC calculation unit block diagram 612
Figure 70. XSPI block diagram for quad configuration 620
Figure 71. SDR read command in octal configuration 621
Figure 72. DTR read in octal-SPI mode with DQS (Macronix mode) example 624
Figure 73. SDR write command in octal-SPI mode example 626
Figure 74. DTR write in octal-SPI mode (Macronix mode) example 627
Figure 75. D0/D1 data ordering in octal-SPI DTR mode (Micron) - Read access 632
Figure 76. OctaRAM read operation with reverse data ordering D1/D0 632
Figure 77. NCS when CKMODE = 0 (T = CLK period) 636
Figure 78. Example of software control of two external memories (octal mode) 637
Figure 79. DLYB block diagram 654
Figure 80. ADC block diagram 662
Figure 81. ADC calibration 665
Figure 82. Calibration factor forcing 666
Figure 83. Enabling/disabling the ADC 667
Figure 84. ADC clock scheme 668
Figure 85. ADC4 connectivity 669
Figure 86. Analog-to-digital conversion time 674
Figure 87. ADC conversion timings 674
Figure 88. Stopping an ongoing conversion 675
Figure 89. Single conversions of a sequence, software trigger 678
Figure 90. Continuous conversion of a sequence, software trigger 679
Figure 91. Single conversions of a sequence, hardware trigger 679
Figure 92. Continuous conversions of a sequence, hardware trigger 680
Figure 93. Data alignment and resolution (oversampling disabled: OVSE = 0) 681
Figure 94. Example of overrun (OVR) 682
Figure 95. Wait conversion mode (continuous mode, software trigger) 684
Figure 96. Auto-off mode state diagram 686
Figure 97. ADC behavior with WAIT = 0 and AUTOFF = 1 686

Figure 98. ADC behavior with WAIT = 1 and AUTOFF = 1 . . . 687
Figure 99. Autonomous mode state diagram . . . 688
Figure 100. Analog watchdog guarded area . . . 689
Figure 101. ADC_AWDx_OUT signal generation . . . 690
Figure 102. ADC_AWDx_OUT signal generation (AWDx flag not cleared by software) . . . 691
Figure 103. ADC_AWDx_OUT signal generation (on a single channel) . . . 691
Figure 104. Analog watchdog threshold update . . . 692
Figure 105. 20-bit to 16-bit result truncation . . . 692
Figure 106. Numerical example with 5-bits shift and rounding . . . 693
Figure 107. Triggered oversampling mode (TOVS bit = 1) . . . 695
Figure 108. Temperature sensor and VREFINT channel block diagram . . . 696
Figure 109. RNG block diagram . . . 723
Figure 110. NIST SP800-90B entropy source model . . . 724
Figure 111. RNG initialization overview . . . 727
Figure 112. AES block diagram . . . 743
Figure 113. Encryption/ decryption typical usage . . . 745
Figure 114. Typical operation with authentication . . . 747
Figure 115. Example of suspend mode management . . . 748
Figure 116. ECB encryption . . . 749
Figure 117. ECB decryption . . . 749
Figure 118. CBC encryption . . . 750
Figure 119. CBC decryption . . . 750
Figure 120. Message construction in CTR mode . . . 752
Figure 121. CTR encryption . . . 753
Figure 122. Message construction in GCM . . . 754
Figure 123. GCM authenticated encryption . . . 756
Figure 124. Message construction in GMAC mode . . . 759
Figure 125. GMAC authentication mode . . . 760
Figure 126. Message construction in CCM mode . . . 761
Figure 127. CCM mode authenticated encryption . . . 762
Figure 128. 128-bit block construction according to the data type . . . 767
Figure 129. HASH block diagram . . . 786
Figure 130. Message data swapping feature . . . 788
Figure 131. HASH suspend/resume mechanism . . . 793
Figure 132. OTFDEC block diagram . . . 807
Figure 133. Typical OTFDEC use in a SoC . . . 808
Figure 134. AES CTR decryption flow . . . 809
Figure 135. OTFDEC flow control overview (dual burst read request) . . . 810
Figure 136. PKA block diagram . . . 830
Figure 137. General-purpose timer block diagram . . . 864
Figure 138. Counter timing diagram with prescaler division change from 1 to 2 . . . 869
Figure 139. Counter timing diagram with prescaler division change from 1 to 4 . . . 869
Figure 140. Counter timing diagram, internal clock divided by 1 . . . 870
Figure 141. Counter timing diagram, internal clock divided by 2 . . . 871
Figure 142. Counter timing diagram, internal clock divided by 4 . . . 871
Figure 143. Counter timing diagram, internal clock divided by N . . . 872
Figure 144. Counter timing diagram, Update event when ARPE = 0 (TIMx_ARR not preloaded) . . . 872
Figure 145. Counter timing diagram, Update event when ARPE = 1 (TIMx_ARR preloaded) . . . 873
Figure 146. Counter timing diagram, internal clock divided by 1 . . . 874
Figure 147. Counter timing diagram, internal clock divided by 2 . . . 875
Figure 148. Counter timing diagram, internal clock divided by 4 . . . 875
Figure 149. Counter timing diagram, internal clock divided by N . . . 876

Figure 150. Counter timing diagram, Update event . . . 876
Figure 151. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 . . . 878
Figure 152. Counter timing diagram, internal clock divided by 2 . . . 878
Figure 153. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36 . . . 879
Figure 154. Counter timing diagram, internal clock divided by N . . . 879
Figure 155. Counter timing diagram, Update event with ARPE = 1 (counter underflow) . . . 880
Figure 156. Counter timing diagram, Update event with ARPE = 1 (counter overflow) . . . 881
Figure 157. Control circuit in normal mode, internal clock divided by 1 . . . 882
Figure 158. tim_ti2 external clock connection example . . . 882
Figure 159. Control circuit in external clock mode 1 . . . 883
Figure 160. External trigger input block . . . 884
Figure 161. Control circuit in external clock mode 2 . . . 885
Figure 162. Capture/compare channel (example: channel 1 input stage) . . . 885
Figure 163. Capture/compare channel 1 main circuit . . . 886
Figure 164. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4) . . . 886
Figure 165. PWM input mode timing . . . 889
Figure 166. Output compare mode, toggle on tim_oc1 . . . 891
Figure 167. Edge-aligned PWM waveforms (ARR = 8) . . . 892
Figure 168. Center-aligned PWM waveforms (ARR = 8) . . . 893
Figure 169. Dithering principle . . . 894
Figure 170. Data format and register coding in dithering mode . . . 895
Figure 171. PWM resolution vs frequency (16-bit mode) . . . 896
Figure 172. PWM resolution vs frequency (32-bit mode) . . . 896
Figure 173. PWM dithering pattern . . . 897
Figure 174. Dithering effect on duty cycle in center-aligned PWM mode . . . 898
Figure 175. Generation of two phase-shifted PWM signals with 50% duty cycle . . . 900
Figure 176. Combined PWM mode on channels 1 and 3 . . . 901
Figure 177. OCREF_CLR input selection multiplexer . . . 902
Figure 178. Clearing TIMx tim_ocxref . . . 902
Figure 179. Example of One-pulse mode . . . 903
Figure 180. Retriggerable one-pulse mode . . . 905
Figure 181. Pulse generator circuitry . . . 906
Figure 182. Pulse generation on compare event, for edge-aligned and encoder modes . . . 906
Figure 183. Extended pulse width in case of concurrent triggers . . . 907
Figure 184. Example of counter operation in encoder interface mode . . . 909
Figure 185. Example of encoder interface mode with tim_ti1fp1 polarity inverted . . . 909
Figure 186. Quadrature encoder counting modes . . . 910
Figure 187. Direction plus clock encoder mode . . . 911
Figure 188. Directional clock encoder mode (CC1P = CC2P = 0) . . . 912
Figure 189. Directional clock encoder mode (CC1P = CC2P = 1) . . . 912
Figure 190. Index gating options . . . 914
Figure 191. Jittered Index signals . . . 914
Figure 192. Index generation for IPOS[1:0] = 11 . . . 915
Figure 193. Counter reading with index gated on channel A (IPOS[1:0] = 11) . . . 915
Figure 194. Counter reading with index ungated (IPOS[1:0] = 00) . . . 916
Figure 195. Counter reading with index gated on channel A and B . . . 916
Figure 196. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11) . . . 917
Figure 197. Counter reset Narrow index pulse (closer view, ARR = 0x07) . . . 918
Figure 198. Index behavior in x1 and x2 mode (IPOS[1:0] = 01) . . . 919
Figure 199. Directional index sensitivity . . . 919
Figure 200. Counter reset as function of FIDX bit setting . . . 920
Figure 201. Index blanking . . . 920

Figure 202. Index behavior in clock + direction mode, IPOS[0] = 1 . . . 921
Figure 203. Index behavior in directional clock mode, IPOS[0] = 1 . . . 921
Figure 204. State diagram for quadrature encoded signals . . . 922
Figure 205. Up-counting encoder error detection . . . 923
Figure 206. Down-counting encode error detection . . . 924
Figure 207. Encoder mode change with preload transferred on update (SMSPS = 0) . . . 925
Figure 208. Control circuit in reset mode . . . 927
Figure 209. Control circuit in gated mode . . . 928
Figure 210. Control circuit in trigger mode . . . 928
Figure 211. Control circuit in external clock mode 2 + trigger mode . . . 930
Figure 212. Master/Slave timer example . . . 930
Figure 213. Master/slave connection example with 1 channel only timers . . . 931
Figure 214. Gating TIM_slv with tim_oc1ref of TIM_mstr . . . 932
Figure 215. Gating TIM_slv with Enable of TIM_mstr . . . 933
Figure 216. Triggering TIM_slv with update of TIM_mstr . . . 934
Figure 217. Triggering TIM_slv with Enable of TIM_mstr . . . 934
Figure 218. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input . . . 935
Figure 219. TIM16/TIM17 block diagram . . . 976
Figure 220. Counter timing diagram with prescaler division change from 1 to 2 . . . 980
Figure 221. Counter timing diagram with prescaler division change from 1 to 4 . . . 980
Figure 222. Counter timing diagram, internal clock divided by 1 . . . 982
Figure 223. Counter timing diagram, internal clock divided by 2 . . . 982
Figure 224. Counter timing diagram, internal clock divided by 4 . . . 983
Figure 225. Counter timing diagram, internal clock divided by N . . . 983
Figure 226. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded) . . . 984
Figure 227. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded) . . . 985
Figure 228. Update rate examples depending on mode and TIMx_RCR register settings . . . 986
Figure 229. Control circuit in normal mode, internal clock divided by 1 . . . 987
Figure 230. tim_ti2 external clock connection example . . . 987
Figure 231. Control circuit in external clock mode 1 . . . 988
Figure 232. Capture/compare channel (example: channel 1 input stage) . . . 989
Figure 233. Capture/compare channel 1 main circuit . . . 989
Figure 234. Output stage of capture/compare channel (channel 1) . . . 990
Figure 235. Output compare mode, toggle on tim_oc1 . . . 993
Figure 236. Edge-aligned PWM waveforms (ARR = 8) . . . 994
Figure 237. Dithering principle . . . 995
Figure 238. Data format and register coding in dithering mode . . . 995
Figure 239. PWM resolution vs frequency . . . 996
Figure 240. PWM dithering pattern . . . 997
Figure 241. Complementary output with symmetrical dead-time insertion . . . 999
Figure 242. Asymmetrical deadtime . . . 999
Figure 243. Dead-time waveforms with delay greater than the negative pulse . . . 1000
Figure 244. Dead-time waveforms with delay greater than the positive pulse . . . 1000
Figure 245. Break circuitry overview . . . 1002
Figure 246. Output behavior in response to a break event on tim_brk . . . 1004
Figure 247. Output redirection . . . 1006
Figure 248. tim_ocref_clr input selection multiplexer . . . 1007
Figure 249. 6-step generation, COM example (OSSR = 1) . . . 1008
Figure 250. Example of one pulse mode . . . 1009
Figure 251. LPTIM block diagram(1) . . . 1042

Figure 252. Glitch filter timing diagram . . . 1046
Figure 253. LPTIM output waveform, single-counting mode configuration when repetition register content is different than zero (with PRELOAD = 1) . . . 1047
Figure 254. LPTIM output waveform, single-counting mode configuration and Set-once mode activated (WAVE bit is set) . . . 1048
Figure 255. LPTIM output waveform, Continuous counting mode configuration . . . 1048
Figure 256. Waveform generation . . . 1050
Figure 257. Encoder mode counting sequence . . . 1054
Figure 258. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1) . . . 1055
Figure 259. Capture/compare input stage (channel 1) . . . 1056
Figure 260. Capture/compare output stage (channel 1) . . . 1056
Figure 261. Edge-aligned PWM mode (PRELOAD = 1) . . . 1058
Figure 262. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0) . . . 1059
Figure 263. PWM mode with immediate update versus preloaded update . . . 1060
Figure 264. IRTIM internal hardware connections with TIM16 and TIM17 . . . 1086
Figure 265. Independent watchdog block diagram . . . 1088
Figure 266. Reset timing due to timeout . . . 1090
Figure 267. Reset timing due to refresh in the not allowed area . . . 1091
Figure 268. Changing PR, RL, and performing a refresh(1) . . . 1092
Figure 269. Independent watchdog interrupt timing diagram . . . 1094
Figure 270. RTC block diagram . . . 1103
Figure 271. TAMP block diagram . . . 1157
Figure 272. Backup registers protection zones . . . 1162
Figure 273. Tamper sampling with precharge pulse . . . 1166
Figure 274. Low level detection with precharge and filtering . . . 1167
Figure 275. Active tamper filtering . . . 1169
Figure 276. Block diagram . . . 1202
Figure 277. I²C-bus protocol . . . 1205
Figure 278. Setup and hold timings . . . 1207
Figure 279. I2C initialization flow . . . 1209
Figure 280. Data reception . . . 1210
Figure 281. Data transmission . . . 1211
Figure 282. Target initialization flow . . . 1214
Figure 283. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0 . . . 1216
Figure 284. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1 . . . 1217
Figure 285. Transfer bus diagrams for I2C target transmitter (mandatory events only) . . . 1218
Figure 286. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0 . . . 1219
Figure 287. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1 . . . 1220
Figure 288. Transfer bus diagrams for I2C target receiver (mandatory events only) . . . 1220
Figure 289. Controller clock generation . . . 1222
Figure 290. Controller initialization flow . . . 1224
Figure 291. 10-bit address read access with HEAD10R = 0 . . . 1224
Figure 292. 10-bit address read access with HEAD10R = 1 . . . 1225
Figure 293. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes . . . 1226
Figure 294. Transfer sequence flow for I2C controller transmitter, N > 255 bytes . . . 1227
Figure 295. Transfer bus diagrams for I2C controller transmitter (mandatory events only) . . . 1228
Figure 296. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes . . . 1230
Figure 297. Transfer sequence flow for I2C controller receiver, N > 255 bytes . . . 1231
Figure 298. Transfer bus diagrams for I2C controller receiver

(mandatory events only) . . . 1232
Figure 299. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$ . . . 1236
Figure 300. Transfer sequence flow for SMBus target transmitter N bytes + PEC . . . 1239
Figure 301. Transfer bus diagram for SMBus target transmitter (SBC = 1) . . . 1240
Figure 302. Transfer sequence flow for SMBus target receiver N bytes + PEC . . . 1241
Figure 303. Bus transfer diagrams for SMBus target receiver (SBC = 1) . . . 1242
Figure 304. Bus transfer diagrams for SMBus controller transmitter . . . 1243
Figure 305. Bus transfer diagrams for SMBus controller receiver . . . 1245
Figure 306. USART block diagram . . . 1271
Figure 307. Word length programming . . . 1275
Figure 308. Configurable stop bits . . . 1277
Figure 309. TC/TXE behavior when transmitting . . . 1279
Figure 310. Start bit detection when oversampling by 16 or 8 . . . 1280
Figure 311. usart_ker_ck clock divider block diagram . . . 1283
Figure 312. Data sampling when oversampling by 16 . . . 1284
Figure 313. Data sampling when oversampling by 8 . . . 1285
Figure 314. Mute mode using Idle line detection . . . 1292
Figure 315. Mute mode using address mark detection . . . 1293
Figure 316. Break detection in LIN mode (11-bit break length - LBDL bit is set) . . . 1296
Figure 317. Break detection in LIN mode vs. Framing error detection . . . 1297
Figure 318. USART example of synchronous master transmission . . . 1298
Figure 319. USART data clock timing diagram in synchronous master mode (M bits = 00) . . . 1298
Figure 320. USART data clock timing diagram in synchronous master mode (M bits = 01) . . . 1299
Figure 321. USART data clock timing diagram in synchronous slave mode (M bits = 00) . . . 1300
Figure 322. ISO 7816-3 asynchronous protocol . . . 1302
Figure 323. Parity error detection using the 1.5 stop bits . . . 1304
Figure 324. IrDA SIR ENDEC block diagram . . . 1308
Figure 325. IrDA data modulation (3/16) - normal mode . . . 1308
Figure 326. Transmission using DMA . . . 1310
Figure 327. Reception using DMA . . . 1311
Figure 328. Hardware flow control between two USARTs . . . 1311
Figure 329. RS232 RTS flow control . . . 1312
Figure 330. RS232 CTS flow control . . . 1313
Figure 331. LPUART block diagram . . . 1359
Figure 332. LPUART word length programming . . . 1363
Figure 333. Configurable stop bits . . . 1365
Figure 334. TC/TXE behavior when transmitting . . . 1367
Figure 335. lpuart_ker_ck clock divider block diagram . . . 1371
Figure 336. Mute mode using Idle line detection . . . 1375
Figure 337. Mute mode using address mark detection . . . 1376
Figure 338. Transmission using DMA . . . 1378
Figure 339. Reception using DMA . . . 1379
Figure 340. Hardware flow control between two LPUARTs . . . 1380
Figure 341. RS232 RTS flow control . . . 1380
Figure 342. RS232 CTS flow control . . . 1381
Figure 343. SPI block diagram . . . 1415
Figure 344. Full-duplex single master/ single slave application . . . 1418
Figure 345. Half-duplex single master/ single slave application . . . 1419
Figure 346. Simplex single master / single slave application

(master in transmit-only / slave in receive-only mode) . . . 1420
Figure 347. Master and three independent slaves connected in star topology . . . 1421
Figure 348. Master and three slaves connected in circular (daisy chain) topology . . . 1423
Figure 349. Multimaster application . . . 1424
Figure 350. Scheme of NSS control logic . . . 1426
Figure 351. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0) . . . 1426
Figure 352. NSS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0) . . . 1427
Figure 353. Data clock timing diagram . . . 1429
Figure 354. TI mode transfer . . . 1440
Figure 355. Optional configurations of the slave behavior when an underrun condition is detected . 1442
Figure 356. SAI functional block diagram . . . 1465
Figure 357. Audio frame . . . 1468
Figure 358. FS role is start of frame + channel side identification (FSDEF = TRIS = 1) . . . 1470
Figure 359. FS role is start of frame (FSDEF = 0) . . . 1471
Figure 360. Slot size configuration with FBOFF = 0 in SAI_xSLOTR . . . 1472
Figure 361. First bit offset . . . 1472
Figure 362. Audio block clock generator overview . . . 1474
Figure 363. PDM typical connection and timing . . . 1478
Figure 364. Detailed PDM interface block diagram . . . 1479
Figure 365. Start-up sequence . . . 1480
Figure 366. SAI_ADR format in TDM mode, 32-bit slot width . . . 1481
Figure 367. SAI_ADR format in TDM mode, 16-bit slot width . . . 1482
Figure 368. SAI_ADR format in TDM mode, 8-bit slot width . . . 1483
Figure 369. AC’97 audio frame . . . 1486
Figure 370. SPDIF format . . . 1487
Figure 371. SAI_xDR register ordering . . . 1488
Figure 372. Data companding hardware in an audio block in the SAI . . . 1491
Figure 373. Tristate strategy on SD output line on an inactive slot . . . 1493
Figure 374. Tristate on output data line in a protocol like I2S . . . 1494
Figure 375. Overrun detection error . . . 1495
Figure 376. FIFO underrun event . . . 1495
Figure 377. USB peripheral block diagram . . . 1529
Figure 378. Packet buffer areas with examples of buffer description table locations . . . 1536
Figure 379. Block diagram of debug support infrastructure . . . 1576
Figure 380. JTAG TAP state machine . . . 1580
Figure 381. AP0: CoreSight topology . . . 1598
Figure 382. CPU CoreSight topology . . . 1607
Figure 383. TPIU architecture . . . 1676
Figure 384. Embedded cross trigger . . . 1688

1 Documentation conventions

1.1 General information

The STM32WBA2 devices embed an Arm®(a) Cortex®-M33 with Arm® TrustZone®, MPU, DSP and FPU core.

1.2 List of abbreviations for registers

The following abbreviations(b) are used in register descriptions:

read/write (rw)	Software can read and write to this bit.
read-only (r)	Software can only read this bit.
write-only (w)	Software can only write to this bit. Reading this bit returns the reset value.
read/clear write0 (rc_w0)	Software can read as well as clear this bit by writing 0. Writing 1 has no effect on the bit value.
read/clear write1 (rc_w1)	Software can read as well as clear this bit by writing 1. Writing 0 has no effect on the bit value.
read/clear write (rc_w)	Software can read as well as clear this bit by writing to the register. The value written to this bit is not important.
read/clear by read (rc_r)	Software can read this bit. Reading this bit automatically clears it to 0. Writing this bit has no effect on the bit value.
read/set by read (rs_r)	Software can read this bit. Reading this bit automatically sets it to 1. Writing this bit has no effect on the bit value.
read/set (rs)	Software can read as well as set this bit. Writing 0 has no effect on the bit value.
read/write once (rwo)	Software can only write once to this bit and can also read it at any time. Only a reset can return the bit to its reset value.
toggle (t)	The software can toggle this bit by writing 1. Writing 0 has no effect.
read-only write trigger (rt_w1)	Software can read this bit. Writing 1 triggers an event but has no effect on the bit value.
Reserved (Res.)	Reserved bit, must be kept at reset value.

a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

b. This is an exhaustive list of all abbreviations applicable to STMicroelectronics microcontrollers, some of them may not be used in the current document.

1.3 Register reset value

Bits (binary notation) or bits nibbles (hexadecimal notation) of which the reset value is undefined are marked as X.

Bits (binary notation) or bits nibbles (hexadecimal notation) of which the reset value is unmodified are marked as U.

1.4 Glossary

This section gives a brief definition of acronyms and abbreviations used in this document:

- **Word**: data of 32-bit length.
- **Half-word**: data of 16-bit length.
- **Byte**: data of 8-bit length.
- **AHB**: advanced high-performance bus.
- **APB**: advanced peripheral bus.

1.5 Availability of peripherals

For availability of peripherals and their number across all sales types, refer to the particular device datasheet.

2 Memory and bus architecture

The device architecture relies on an Arm® Cortex®-M33 core, optimized for execution through an instruction cache that accesses the embedded flash memory.

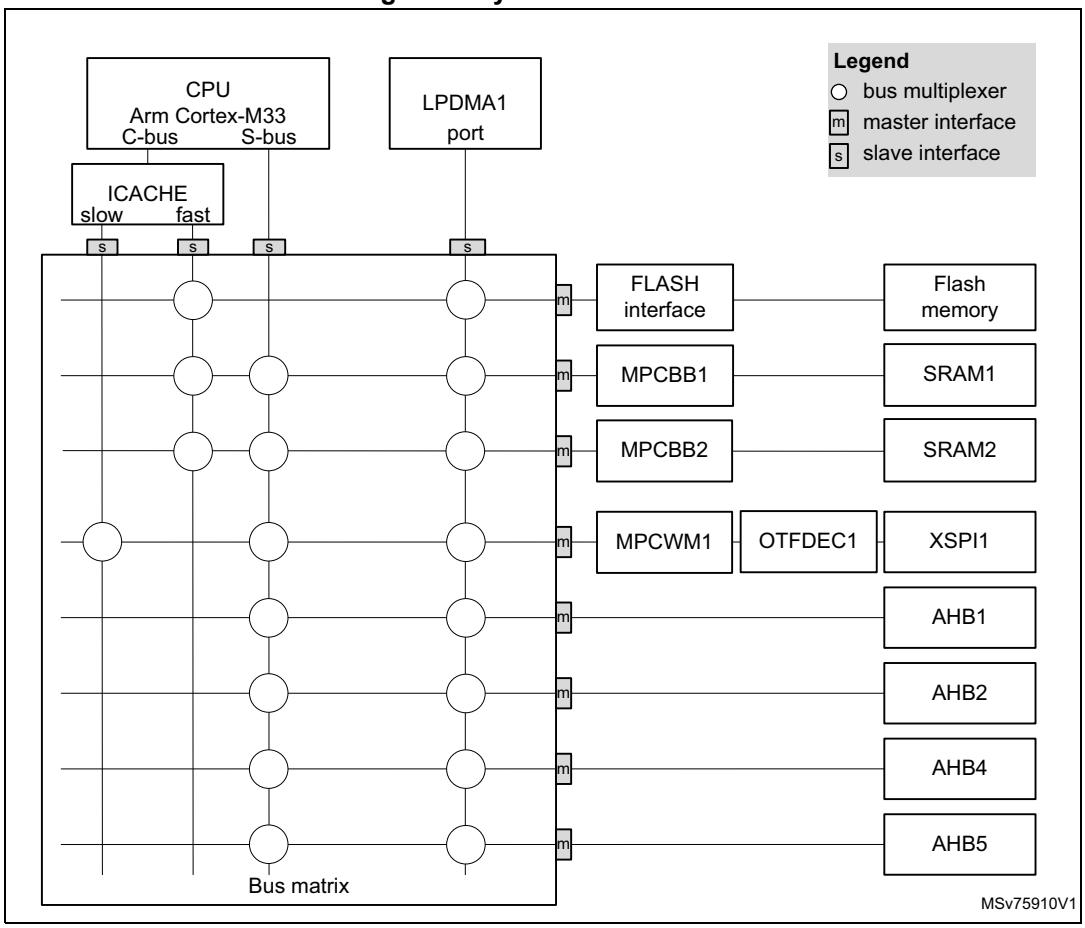
2.1 System architecture

The architecture features a 32-bit multilayer AHB bus matrix that interconnects:

- Masters:
  - CPU core C-bus through ICACHE fast port, connecting to flash and SRAM memory
  - CPU core C-bus through ICACHE slow port remapping, connecting to external memory
  - CPU core S-bus
  - LPDMA1
- Slaves:
  - Internal flash memory
  - Internal SRAM1
  - Internal SRAM2
  - AHB1 peripherals including AHB to APB bridges and APB peripherals (connected to APB1 and APB2)
  - AHB2 peripherals
  - AHB4 peripherals including AHB to APB bridges and APB peripherals (connected to APB7)
  - AHB5 with the 2.4 GHz RADIO peripheral
  - AHB5 with PTACONV peripheral

The bus matrix provides access from a master to a slave, enabling concurrent access and efficient operation even when several high-speed peripherals work simultaneously. The architecture is shown in *Figure 1*.

Figure 1. System architecture



2.1.1 CPU C-bus

This bus connects the C-bus of the CPU to the internal flash memory and to the bus matrix. This bus is used for instruction fetching and data access to the internal memories mapped in the code region. This bus targets the internal flash memory, internal SRAM (SRAM1 and SRAM2), and XSPI1 bank through the ICACHE address remap function.

2.1.2 CPU S-bus

This bus connects the system bus of the CPU to the bus matrix, and it is used by the core to access data located in a peripheral or SRAM area. This bus targets the internal SRAM (SRAM1 and SRAM2), XSPI1 bank, AHB1 peripherals including the APB1 and APB2 peripherals, AHB2, AHB4 peripherals including the APB7, and AHB5 peripherals.

2.1.3 LPDMA1-bus

The bus connects the AHB master interface of the LPDMA1 to the bus matrix. This targets the internal flash memory, the internal SRAM (SRAM1 and SRAM2), XSPI1 bank, AHB1 peripherals including the APB1 and APB2 peripherals, AHB2 peripherals, AHB4 peripherals including the APB7, and AHB5 peripherals.

2.1.4 Bus matrix

The bus matrix manages the access arbitration (based on fixed priority) between masters, and features a bus multiplexer used to connect each master to a given slave with no latency.

Table 1. Bus matrix acces arbitration

Master	Priority
Cpu core S-bus	1 - highest
ICACHE slow port	2
ICACHE fast port	3
LPDMA1	4 - lowest

2.1.5 AHB/APB bridges

The three bridges, AHB1 to APB1, AHB1 to ABP2, and AHB4 to APB7, provide full synchronous connections between the AHB and the APB buses, resulting in the flexible selection of the peripheral frequency.

Refer to *Section 2.3.2: Memory map and register boundary addresses* for the address mapping of the peripherals connected to these bridges.

After each device reset, the clock of peripherals having a xxEN bit in the RCC is disabled. Before using a peripheral, its clock must be enabled in the RCC_AHBxENR and RCC_APBxENR registers.

Note: When a 16- or 8-bit access operation is performed on an APB register, it is transformed into a 32-bit access operation: the bridge duplicates the 16- or 8-bit data to feed the 32-bit vector.

The 2.4 GHz RADIO peripheral AHB5 is a semisynchronous bus, connected through a bridge to the bus matrix.

2.2 TrustZone® security architecture

The security architecture is based on Arm® TrustZone® with the Armv8_M mainline extension.

The TZEN option bit in the FLASH_OPTR register activates TrustZone® security.

When TrustZone® is enabled, the SAU (security attribution unit) and IDAU (implementation-defined attribution unit) define the access permissions based on secure and nonsecure states.

- SAU: up to eight SAU configurable regions are available for security attribution.
- IDAU: provides a first memory partition as nonsecure or nonsecure callable attributes. The IDAU memory map partition is not configurable and fixed by hardware implementation (refer to *Figure 2: Memory map*). It is then combined with the results from the SAU security attribution, and the highest security state is selected.

Based on IDAU security attribution, the flash memory, system SRAMs, and peripheral memory space are aliased twice for secure and nonsecure states. However, the external memory space is not aliased.

Table 2 shows a typical example of eight SAU regions mapping, based on IDAU regions. The user can split and choose the secure, nonsecure, or NSC regions for external memories according to application requirements.

Table 2. Memory map security attribution example vs. SAU configuration regions(1)

Region description	Address range	IDAU security attribution	SAU security attribution typical configuration	Final security attribution
Reserved	0x0000 0000 to 0x07FF FFFF	Nonsecure	Secure, nonsecure, or NSC	
Code Flash and SRAM	0x0800 0000 to 0x0BFF FFFF	Nonsecure	Nonsecure	
	0x0C00 0000 to 0x0FFF FFFF	NSC	Secure or NSC	
Reserved	0x1000 0000 to 0x17FF FFFF	Nonsecure		
	0x1800 0000 to 0x1FFF FFFF			
SRAM	0x2000 0000 to 0x2FFF FFFFF	Nonsecure		
	0x3000 0000 to 0x3FFF FFFFF	NSC	Secure or NSC	
Peripherals	0x4000 0000 to 0x4FFF FFFFF	Nonsecure	Nonsecure	
	0x5000 0000 to 0x5FFF FFFFF	NSC	Secure or NSC	
External memories	0x6000 0000 to 0xDFFF FFFF	Nonsecure	Secure, nonsecure, or NSC	

1. NSC = nonsecure callable

2.2.1 Default TrustZone® security state

When the TrustZone® security is activated by the TZEN option bit in the FLASH_OPTR register, the default system security state is as detailed below:

- CPU:
  - Core is in a secure state after reset. The boot address must point to a secure area.
- Memory map:
  - SAU is fully secure after reset. Consequently, the entire memory map is fully secure. Up to eight SAU configurable regions are available for security attribution.
- Flash memory:
  - The flash memory security area is defined by watermark user options.
  - Flash memory block-based security attributions are nonsecure after reset.
- SRAMs:
  - SRAM1 and SRAM2 are secure after reset. MPCBBx (block-based memory protection controller) is secure.
- External memories:
  - The XSPI bank is secure after reset. MPCWM (watermark-based memory protection controller) is secure.
- Peripherals (see *Table 3 and Table 4 for a list of securable and TrustZone®-aware peripherals):*
  - Securable peripherals are nonsecure after reset.
  - TrustZone®-aware peripherals are nonsecure after reset. Their secure configuration registers are secure.
- All GPIOs are secure after reset.
- Interrupts:
  - NVIC: all interrupts are secure after reset. NVIC is banked for secure and nonsecure states.
  - TZIC: all illegal access interrupts are disabled after reset (see *Section 5.5: GTZC interrupts*).

2.2.2 TrustZone® peripheral classification

When the TrustZone® security is active, a peripheral can be either the securable or TrustZone®-aware type as follows:

- Securable: peripheral protected by an AHB/APB firewall gate controlled by the TZSC controller to define security properties
- TrustZone®-aware: peripheral connected directly to the AHB or APB bus and implementing specific TrustZone® behavior, such as a subset of registers being secure.

Refer to *Section 5: Global TrustZone® controller (GTZC)* for more details.

Table 3 and *Table 4* list the securable and TrustZone®-aware peripherals within the system.

Table 3. Securable peripherals by TZSC

Bus	Peripheral
AHB5	2.4 GHz RADIO + SEQRAM
	PTACONV

Table 3. Securable peripherals by TZSC (continued)

Bus	Peripheral
AHB4	ADC4
AHB2	XSPI1 and DLYBXS1(1)
	PKA
	RNG
	HASH
	AES
AHB1	ICACHE registers
	CRC
	RAMCFG
APB7	LPTIM1
	I2C3
	LPUART1
	SPI3
APB2	TIM17
	TIM16
	USB(1)
	SAI1
	USART1
APB1	LPTIM2
	I2C1
	IWDG
	TIM2

1. Only available on STM32WBA25xx devices.

Table 4. TrustZone®-aware peripherals

Bus	Peripheral
AHB4	EXTI
	RCC
	PWR
AHB2	OTFDEC1(1)
	GPIOH
	GPIOC
	GPIOB
	GPIOA

Table 4. TrustZone®-aware peripherals (continued)

Bus	Peripheral
AHB1	GTZC1-MPCBB2
	GTZC1-MPCBB1
	GTZC1-TZSC
	GTZC1-TZIC
	FLASH interface
	LPDMA1
APB7	TAMP
	RTC
	SYSCFG

1. Only available on STM32WBA25xx devices.

2.3 Memory organization

2.3.1 Introduction

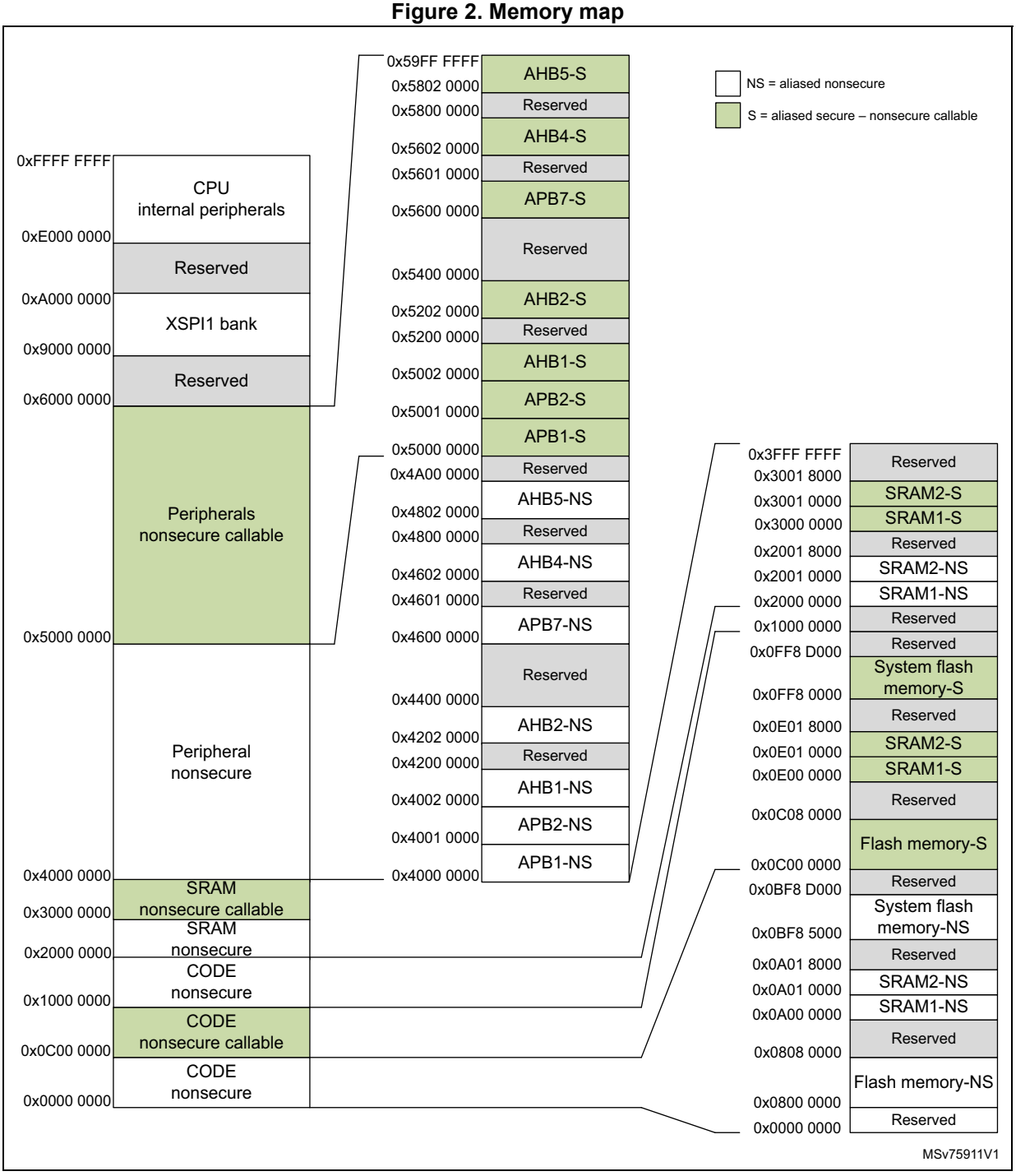
Program memory, data memory, registers and I/O ports are organized within the same linear address space.

The bytes are coded in memory in Little Endian format. The lowest numbered byte in a word is considered the word's least significant byte and the highest numbered byte the most significant.

The addressable memory space is divided into eight main blocks, of 512 Mbytes each.

2.3.2 Memory map and register boundary addresses

Figure 2. Memory map



Any memory area not allocated to on-chip memories and peripherals is considered “Reserved”.

Table 5 gives the boundary addresses of the peripherals available in the device.

!!! LINKS TO BE UPDATED !!!

Table 5. Memory map and peripheral register boundary addresses

Bus	Secure Boundary address(1)	Nonsecure Boundary address(1)	Size (bytes)	Peripheral	Peripheral register map
-	0xA000 0000 - 0xDFFF FFFF	0xA000 0000 - 0xDFFF FFFF	-	Reserved	-
-	0x9000 0000 - 0x9FFF FFFF	0x9000 0000 - 0x9FFF FFFF	256 M	XSPI1 bank(2)	-
-	0x5A00 0000 - 0x8FFF FFFF	0x4A00 0000 - 0x8FFF FFFF	-	Reserved	-
AHB5	0x5803 8400 - 0x59FF FFFF	0x4803 8400 - 0x49FF FFFF	-	Reserved	-
	0x5803 8000 - 0x5803 83FF	0x4803 8000 - 0x4803 83FF	1 K	PTACONV	*Section 10.5.4 on page 281*
	0x5802 A800 - 0x5803 7FFF	0x4802 A800 - 0x4803 7FFF	-	Reserved	-
	0x5802 8000 - 0x5802 A7FF	0x4802 8000 - 0x4802 A7FF	10 K	RXTXRAM	-
	0x5802 1200 - 0x5802 7FFF	0x4802 1200 - 0x4802 7FFF	-	Reserved	-
	0x5802 1000 - 0x5802 11FF	0x4802 1000 - 0x4802 11FF	0.5 K	SEQRAM	-
	0x5802 0000 - 0x5802 0FFF	0x4802 0000 - 0x4802 0FFF	4 K	2.4 GHz RADIO	-
-	0x5800 0000 - 0x5801 FFFF	0x4800 0000 - 0x4801 FFFF	-	Reserved	-
AHB4	0x5602 2400 - 0x57FF FFFF	0x4602 2400 - 0x47FF FFFF	-	Reserved	-
	0x5602 2000 - 0x5602 23FF	0x4602 2000 - 0x4602 23FF	1 K	EXTI	*Section 18.6.15 on page 609*
	0x5602 1400 - 0x5602 1FFF	0x4602 1400 - 0x4602 1FFF	-	Reserved	-
	0x5602 1000 - 0x5602 13FF	0x4602 1000 - 0x4602 13FF	1 K	ADC4	*Section 22.7.18 on page 719*
	0x5602 0C00 - 0x5602 0FFF	0x4602 0C00 - 0x4602 0FFF	1 K	RCC	*Section 12.8.54 on page 449*
	0x5602 0800 - 0x5602 0BFF	0x4602 0800 - 0x4602 0BFF	1 K	PWR	*Section 11.10.25 on page 346*
	0x5602 0000 - 0x5602 07FF	0x4602 0000 - 0x4602 07FF	-	Reserved	-
-	0x5601 0000 - 0x5601 FFFF	0x4601 0000 - 0x4601 FFFF	-	Reserved	-

Table 5. Memory map and peripheral register boundary addresses (continued)

Bus	Secure Boundary address(1)	Nonsecure Boundary address(1)	Size (bytes)	Peripheral	Peripheral register map
APB7	0x5600 8000 - 0x5600 FFFF	0x4600 8000 - 0x4600 FFFF	-	Reserved	-
	0x5600 7C00 - 0x5600 7FFF	0x4600 7C00 - 0x4600 7FFF	1 K	TAMP	*Section 34.6.19 on page 1198*
	0x5600 7800 - 0x5600 7BFF	0x4600 7800 - 0x4600 7BFF	1 K	RTC	*Section 33.6.26 on page 1154*
	0x5600 4800 - 0x5600 77FF	0x4600 4800 - 0x4600 77FF	-	Reserved	-
	0x5600 4400 - 0x5600 47FF	0x4600 4400 - 0x4600 47FF	1 K	LPTIM1	*Section 30.7.16 on page 1084*
	0x5600 2C00 - 0x5600 43FF	0x4600 2C00 - 0x4600 43FF	-	Reserved	-
	0x5600 2800 - 0x5600 2BFF	0x4600 2800 - 0x4600 2BFF	1 K	I2C3	*Section 35.9.13 on page 1266*
	0x5600 2400 - 0x5600 27FF	0x4600 2400 - 0x4600 27FF	1 K	LPUART1	*Section 37.7.15 on page 1410*
	0x5600 2000 - 0x5600 23FF	0x4600 2000 - 0x4600 23FF	1 K	SPI3	*Section 38.8.15 on page 1462*
	0x5600 0800 - 0x5600 1FFF	0x4600 0800 - 0x4600 1FFF	-	Reserved	
	0x5600 0400 - 0x5600 07FF	0x4600 0400 - 0x4600 07FF	1 K	SYSCFG	*Section 14.3.11 on page 516*
	0x5600 0000 - 0x5600 03FF	0x4600 0000 - 0x4600 03FF	-	Reserved	-
-	0x5400 0000 - 0x55FF FFFF	0x4400 0000 - 0x45FF FFFF	-	Reserved	-

Table 5. Memory map and peripheral register boundary addresses (continued)

Bus	Secure Boundary address(1)	Nonsecure Boundary address(1)	Size (bytes)	Peripheral	Peripheral register map
AHB2	0x520D 1800 - 0x53FF FFFF	0x420D 1800 - 0x43FF FFFF	-	Reserved	-
	0x520D 1400 - 0x520D 17FF	0x420D 1400 - 0x420D 17FF	1 K	XSPI1(2)	*Section 20.7.14 on page 651*
	0x520C F400 - 0x520D 13FF	0x420C F400 - 0x420D 13FF	-	Reserved	-
	0x520C F000 - 0x520C F3FF	0x420C F000 - 0x420C F3FF	1 K	DLYBXS1(2)	*Section 21.5.3 on page 658*
	0x520C 5400 - 0x520C EFFF	0x420C 5400 - 0x420C EFFF	-	Reserved	-
	0x520C 5000 - 0x520C 53FF	0x420C 5000 - 0x420C 53FF	1 K	OTFDEC1(2)	*Section 26.6.15 on page 825*
	0x520C 4000 - 0x530C 4FFF	0x420C 4000 - 0x420C 4FFF	-	Reserved	-
	0x520C 3C00 - 0x520C 3FFF	0x420C 3C00 - 0x420C 3FFF	8 K	PKA continue	*Section 27.8.5 on page 861*
	0x520C 2400 - 0x520C 3BFF	0x420C 2400 - 0x420C 3BFF		PKA RAM	
	0x520C 2000 - 0x520C 23FF	0x420C 2000 - 0x420C 23FF		PKA	
	0x520C 0C00 - 0x520C 1FFF	0x420C 0C00 - 0x420C 1FFF	-	Reserved	-
	0x520C 0800 - 0x520C 0BFF	0x420C 0800 - 0x420C 0BFF	1 K	RNG	*Section 23.7.9 on page 741*
	0x520C 0400 - 0x520C 07FF	0x420C 0400 - 0x420C 07FF	1 K	HASH	*Section 25.6.8 on page 804*
	0x520C 0000 - 0x520C 03FF	0x420C 0000 - 0x420C 03FF	1 K	AES	*Section 24.9.21 on page 783*
	0x5202 2000 - 0x520B FFFF	0x4202 2000 - 0x420B FFFF	-	Reserved	-
	0x5202 1C00 - 0x5202 1FFF	0x4202 1C00 - 0x4202 1FFF	1 K	GPIOH	*Section 13.8.1 on page 493*
	0x5202 0C00 - 0x5202 1BFF	0x4202 0C00 - 0x4202 1BFF	-	Reserved	-
	0x5202 0800 - 0x5202 0BFF	0x4202 0800 - 0x4202 0BFF	1 K	GPIOC	*Section 13.7.1 on page 487*
	0x5202 0400 - 0x5202 07FF	0x4202 0400 - 0x4202 07FF	1 K	GPIOB	*Section 13.6.1 on page 477*
	0x5202 0000 - 0x5202 03FF	0x4202 0000 - 0x4202 03FF	1 K	GPIOA	*Section 13.5.1 on page 469*
-	0x5200 0000 - 0x5201 FFFF	0x4200 0000 - 0x4201 FFFF	-	Reserved	-
AHB1	0x5002 3400 - 0x51FF FFFF	0x4002 3400 - 0x41FF FFFF	-	Reserved	-
	0x5003 3000 - 0x5003 33FF	0x4003 3000 - 0x4003 33FF	1 K	GTZC1_MPCBB2	*Section 5.8 on page 158*
	0x5003 2C00 - 0x5003 2FFF	0x4003 2C00 - 0x4003 2FFF	1 K	GTZC1_MPCBB1	*Section 5.8 on page 158*
	0x5003 2800 - 0x5003 2BFF	0x4003 2800 - 0x4003 2BFF	1 K	GTZC1_TZIC	*Section 5.7 on page 142*
	0x5003 2400 - 0x5003 27FF	0x4003 2400 - 0x4003 27FF	1 K	GTZC1_TZSC	*Section 5.6 on page 128*
	0x5003 0800 - 0x5003 23FF	0x4003 0800 - 0x5003 23FF	-	Reserved	-
	0x5003 0400 - 0x5003 07FF	0x4003 0400 - 0x4003 07FF	1 K	ICACHE	*Section 8.7.8 on page 264*
	0x5002 7000 - 0x5003 03FF	0x4002 7000 - 0x4003 03FF	-	Reserved	-
	0x5002 6000 - 0x5002 6FFF	0x4002 6000 - 0x4002 6FFF	4 K	RAMCFG	*Section 6.6.4 on page 170*
	0x5002 3400 - 0x5002 5FFF	0x4002 3400 - 0x4002 5FFF	-	Reserved	-
	0x5002 3000 - 0x5002 33FF	0x4002 3000 - 0x4002 33FF	1 K	CRC	*Section 19.4.6 on page 617*
	0x5002 2400 - 0x5002 2FFF	0x4002 2400 - 0x4002 2FFF	-	Reserved	-
	0x5002 2000 - 0x5002 23FF	0x4002 2000 - 0x4002 23FF	1 K	FLASH interface	*Section 7.9.36 on page 245*
	0x5002 1000 - 0x5002 1FFF	0x4002 1000 - 0x4002 1FFF	-	Reserved	-
	0x5002 0000 - 0x5002 0FFF	0x4002 0000 - 0x4002 0FFF	4 K	LPDMA1	*Section 16.8.16 on page 584*

Table 5. Memory map and peripheral register boundary addresses (continued)

Bus	Secure Boundary address(1)	Nonsecure Boundary address(1)	Size (bytes)	Peripheral	Peripheral register map
APB2	0x5001 6C00 - 0x5001 FFFF	0x4001 6C00 - 0x4001 FFFF	-	Reserved	-
	0x5001 6400 - 0x5001 6BFF	0x4001 6400 - 0x4001 6BFF	2 K	USB SRAM(2)	*Section 40.6.8 on page 1568*
	0x5001 6000 - 0x5001 63FF	0x4001 6000 - 0x4001 63FF	1 K	USB(2)	*Section 40.6.8 on page 1568*
	0x5001 5800 - 0x5001 5FFF	0x4001 5800 - 0x4001 5FFF	-	Reserved	-
	0x5001 5400 - 0x5001 57FF	0x4001 5400 - 0x4001 57FF	1 K	SAI1	*Section 39.6.19 on page 1526*
	0x5001 4C00 - 0x5001 53FF	0x4001 4C00 - 0x4001 53FF	-	Reserved	-
	0x5001 4800 - 0x5001 4BFF	0x4001 4800 - 0x4001 4BFF	1 K	TIM17	*Section 29.7.22 on page 1038*
	0x5001 4400 - 0x5001 47FF	0x4001 4400 - 0x4001 47FF	1 K	TIM16	*Section 29.7.22 on page 1038*
	0x5001 3C00 - 0x5001 43FF	0x4001 3C00 - 0x4001 43FF	-	Reserved	-
	0x5001 3800 - 0x5001 3BFF	0x4001 3800 - 0x4001 3BFF	1 K	USART1	*Section 36.8.17 on page 1355*
	0x5001 0000 - 0x5001 37FF	0x4001 0000 - 0x4001 37FF	-	Reserved	-
APB1	0x5000 9800 - 0x5000 FFFF	0x4000 9800 - 0x4000 FFFF	-	Reserved	-
	0x5000 9400 - 0x5000 97FF	0x4000 9400 - 0x4000 97FF	1 K	LPTIM2	*Section 30.7.16 on page 1084*
	0x5000 5800 - 0x5000 93FF	0x4000 5800 - 0x4000 93FF	-	Reserved	
	0x5000 5400 - 0x5000 57FF	0x4000 5400 - 0x4000 57FF	1 K	I2C1	*Section 35.9.13 on page 1266*
	0x5000 3400 - 0x5000 53FF	0x4000 3400 - 0x4000 53FF	-	Reserved	-
	0x5000 3000 - 0x5000 33FF	0x4000 3000 - 0x4000 33FF	1 K	IWDG	*Section 32.7.7 on page 1101*
	0x5000 0400 - 0x5000 2FFF	0x4000 0400 - 0x4000 2FFF	-	Reserved	-
	0x5000 0000 - 0x5000 03FF	0x4000 0000 - 0x4000 03FF	1 K	TIM2	*Section 28.5.25 on page 972*

Table 5. Memory map and peripheral register boundary addresses (continued)

Bus	Secure Boundary address(1)	Nonsecure Boundary address(1)	Size (bytes)	Peripheral	Peripheral register map
AHB	0x3802 A800 - 0x4FFF FFFF	0x2802 A800 - 0x3FFF FFFF	-	Reserved	-
	0x3802 8000 - 0x3802 A7FF	0x2802 8000 - 0x2802 A7FF	10 K	RXTXSRAM	
	0x3001 8000 - 0x3802 7FFF	0x2001 8000 - 0x2802 7FFF	-	Reserved	-
	0x3001 2000 - 0x3001 7FFF	0x2001 2000 - 0x2001 7FFF	24 K	SRAM2 page 2	-
	0x3001 0000 - 0x3001 1FFF	0x2001 0000 - 0x2001 1FFF	8 K	SRAM2 page 1	
	0x3000 0000 - 0x3000 FFFF	0x2000 0000 - 0x2000 FFFF	64 K	SRAM1	-
	0x0FF9 0000 - 0x2FFF FFFF	0x0BF9 0000 - 0x1FFF FFFF	-	Reserved	-
	0x0FF8 E000 - 0x0FF8 FFFF	0x0BF8 E000 - 0x0BF8 FFFF	16 K	Flash memory user options	*Section 7.4.1 on page 191*
	0x0FF8 D500 - 0x0FF8 DFFF	0x0BF8 D500 - 0x0BF8 DFFF	2.75 K	DESIG	*Section 42.1.11 on page 1724*
	0x0FF8 D200 - 0x0FF8 D4FF	0x0BF8 D200 - 0x0BF8 D4FF	-	Reserved	-
	0x0FF8 D000 - 0x0FF8 D1FF	0x0BF8 D000 - 0x0BF8 D1FF	512	OTP	-
	0x0FF8 5000 - 0x0FF8 CFFF	0x0BF8 5000 - 0x0BF8 CFFF	32 K	Bootloader	-
	0x0FF8 4000 - 0x0FF8 4FFF	0x0BF8 4000 - 0x0BF8 4FFF	4 K	RSS-Lib	-
	0x0FF8 0000 - 0x0FF8 3FFF	0x0BF8 0000 - 0x0BF8 3FFF	16 K	RSS-Boot	-
	0x0E01 8000 - 0x0FF7 FFFF	0x0A01 8000 - 0x0BF7 FFFF	-	Reserved	-
	0x0E01 2000 - 0x0E01 7FFF	0x0A01 2000 - 0x0A01 7FFF	24 K	SRAM2 page 2	-
	0x0E01 0000 - 0x0E01 1FFF	0x0A01 0000 - 0x0A01 1FFF	8 K	SRAM2 page 1	
	0x0E00 0000 - 0x0E00 FFFF	0x0A00 0000 - 0x0A00 FFFF	64 K	SRAM1	-
	0x0C08 0000 - 0x0DFF FFFF	0x0808 0000 - 0x09FF FFFF	-	Reserved	-
	0x0C00 0000 - 0x0C07 FFFF	0x0800 0000 - 0x0807 FFFF	512 K	User flash memory	-
	0x0000 0000 - 0x0BFF FFFF	0x0000 0000 - 0x07FF FFFF	-	Reserved	-

1. Gray shaded fields are reserved.
2. Only available on STM32WBA25xx devices.

2.3.3 Embedded SRAM

The devices feature 96-Kbyte SRAMs:

- SRAM1: 64 Kbytes
- SRAM2: 32 Kbytes in two pages (8 Kbytes and 24 Kbytes)

The SRAMs can be accessed as bytes, half-words (16 bits), or full words (32 bits). These memories can be addressed by both the CPU and DMA.

The CPU can access the SRAMs through the system bus or using the ICACHE through the C-bus, depending on the access address.

When TrustZone® security is enabled, all SRAMs are secure after reset. The SRAM can be programmed as nonsecure with a block granularity. For more details, refer to *Section 5: Global TrustZone® controller (GTZC)*.

SRAM features are detailed in *Section 6.3.1: Internal SRAM features*.

2.3.4 Flash memory overview

The flash memory is composed of two distinct physical areas:

- The main flash memory block, containing the application program and user data.
- The information block, which is composed of the following parts:
  - Option bytes for hardware and memory protection user configuration
  - System memory containing STMicroelectronics proprietary code
  - OTP (one-time programmable) area

The FLASH interface implements instruction access and data access based on the AHB protocol. It also implements the necessary logic to carry out the flash memory operations (program/erase) that are controlled through the FLASH registers, as well as security access control features. Refer to *Section 7: Embedded flash memory (FLASH)* for more details.

3 Boot modes

At startup, a BOOT0 pin, and the NBOOT0, NSWBOOT0, NSBOOTADDx/SECBOOTADD0, and TZEN option bytes are used to select the boot memory address, which includes:

- Boot from any address in the user flash memory
- Boot from system memory (bootloader)
- Boot from any address in the embedded SRAM
- Boot from root security service (RSS)

The BOOT0 value may come from the PH3-BOOT0 pin or from the option bit NBOOT0, depending on the value of a user option bit to free the GPIO pad, if needed.

The bootloader, located in the system memory, is used to program the flash memory by using USART, I2C, or SPI in device mode.

Table 6 details the boot modes when TrustZone® is disabled, and *Table 7* when enabled.

Table 6. Boot modes when TrustZone® is disabled (TZEN = 0)

BOOT0(1)	Boot address option bytes selection		Boot initial VTOR_NS
	Nonsecure	ST programmed default value	
0	NSBOOTADD0	0x0800 0000 - User flash memory	NSBOOTADD0
1	NSBOOTADD1	0x0BF8 5000 - Bootloader	NSBOOTADD1

1. BOOT0 is either not NBOOT0 when NSWBOOT0 = 0 or pin PH3-BOOT0 when NSWBOOT0 = 1

When TrustZone® is enabled by setting the TZEN option bit, the boot space must be in the secure area. The SECBOOTADD0 option bytes are used to select the boot secure memory address.

A unique boot entry option can be selected by setting the BOOT_LOCK option bit. In this case, all other boot options are ignored.

Table 7. Boot modes when TrustZone® is enabled (TZEN = 1)

BOOT_LOCK	BOOT0(1)	RSS command	Boot address option bytes selection				Boot initial VTOR_S
			Secure	ST programmed default value	Nonsecure	ST programmed default value	
0	0	0	SECBOOTADD0	0x0C00 0000 - User flash	NSBOOTADD0	0x0800 0000(2)	SECBOOTADD0
	1	0	N/A	0x0FF8 0000 - RSS	NSBOOTADD1	0x0BF8 5000	0x0FF8 0000
	0	≠ 0	N/A	0x0FF8 0000 - RSS	NSBOOTADD0	0x0800 0000(2)	0x0FF8 0000
	1				NSBOOTADD1	0x0BF8 5000	
1	0	x	SECBOOTADD0	0x0C00 0000 - User flash	NSBOOTADD0	0x0800 0000(2)	SECBOOTADD0
	1				NSBOOTADD1	0x0BF8 5000(3)	

1. BOOT0 is either not NBOOT0 when NSWBOOT0 = 0 or pin PH3-BOOT0 when NSWBOOT0 = 1
2. The default NSBOOTADD0 points to a secure flash memory area, at boot privileged software must write Cortex®-M33 VTOR_NS to point to a nonsecure area.

3. The default NSBOOTADD1 points to the bootloader, at boot privileged software must write Cortex®-M33 VTOR_NS to point to a nonsecure user area.

The boot address option bytes are used to program any boot memory address. However, the allowed address space depends on the flash memory read protection (RDP) level.

If the programmed boot memory address is outside of the allowed memory-mapped area when TZEN = 0 and the RDP level is 2, or when TZEN = 1 and the RDP level is 0.5 or more, the default boot fetch address is forced either in the secure flash memory or the nonsecure flash memory, depending on the TrustZone® security option, as described in *Table 8*.

Table 8. Boot space versus RDP protection

RDP	TZEN = 1(1)	TZEN = 0
	Valid address range SECBOOTADD	Valid address range NSBOOTADDx
0	Any boot address	Any boot address
0.5	Boot address only in RSS 0x0FF8 0000 or in secure flash memory: 0x0C00 0000- 0x0C07 FFFF. Otherwise boot address forced to RSS: 0x0FF8 0000.	N/A
1		Any boot address
2		Boot address only in flash memory: 0x0800 0000 - 0x0807 FFFF. Otherwise boot address forced to 0x0800 0000.

1. The initial NSBOOTADDx can point to a secure area. At boot, privileged software must write Cortex®-M33 VTOR_NS to point to a nonsecure user area.

The BOOT0 value (either coming from the pin or the user option) is latched upon reset release. It is up to the user to set the BOOT0 or NBOOT0 values to select the required boot mode.

The BOOT0 pin or NBOOT0 user option (depending on NSWBOOT0 in FLASH_OPTR) is also resampled when exiting standby modes. Consequently, the BOOT0 pin or user option must be kept in the required boot mode configuration in standby modes. After the startup delay, the selection of the boot area is done before releasing the processor reset.

PH3-BOOT0 GPIO is configured as follows:

- Input mode during the complete reset phase if the NSWBOOT0 option bit is set in FLASH_OPTR, automatically switching to analog mode after reset is released (BOOT0 pin).
- Input mode from the reset phase to the completion of the option byte loading if the NSWBOOT0 bit is cleared in FLASH_OPTR (BOOT0 value coming from the user option), automatically switches to analog mode even if the reset phase is not complete.

Embedded bootloader

The embedded bootloader is located in the system memory and is programmed by STMicroelectronics during production. Refer to *AN2606: STM32 microcontroller system memory boot mode*.

Embedded root security services (RSS)

The embedded RSS are located in the secure information block, programmed by STMicroelectronics during production.

4 System security

The device is designed with a comprehensive set of security features, some of which are based on the standard Arm® TrustZone® technology.

These security features simplify the process of evaluating IoT devices against security standards. They also significantly reduce the cost and complexity of software development for OEM and third-party developers, by facilitating the re-use, improving the interoperability, and minimizing the API fragmentation.

This section explains the different security features available on the device.

4.1 Key security features

- Resource isolation using privileged mode and Armv8-M mainline security extension of Cortex-M33, extended to securable I/Os, memories, and peripherals
- Secure boot thanks to the unique boot entry feature and hide-protect area (HDP) mechanism
- Secure storage, featuring:
  - Nonvolatile on-chip secure storage, protected with secure and HDP areas
  - Volatile secure storage, automatically erased in case of tamper
  - Write-only key registers in the AES engines
  - Device 96-bit unique ID and JTAG 32-bit device-specific ID
- General-purpose cryptographic acceleration
  - AES 256-bit engine, supporting ECB, CBC, CTR, GCM, and CCM chaining modes
  - HASH processor, supporting SHA-1 checksums and SHA-2 secure hash
  - Public key accelerator (PKA) for RSA/DH (up to 4096 bits) and ECC (up to 640 bits), implementing side-channel countermeasures, and mitigations when manipulating secrets
  - True random number generator (RNG), NIST SP800-90B precertified
- On-the-fly decryption of encrypted image stored in external memory connected through OTFDEC.
  - Can be used to encrypt the image using device unique secret keys
  - Automatic key erase in case of tamper
- Flexible life cycle scheme with readout protection (RDP), including support for product decommissioning (autoerase)
  - Debug protection, depending on the RDP level
  - Optional password-based RDP level regressions, including for RDP Level 2
- Protected firmware distribution scheme, using TrustZone and RDP Level 0.5
- Active tamper and protection
  - Four active inputs, four active output tamper pins, available in all power modes, except Standby

4.2 Secure boot

Secure boot is an immutable code that is always executed after a system reset. As a root of trust, this code checks the device static protections and activates available device runtime protections, reducing the risk that invalid or malicious code runs on the platform. As root of trust, the secure boot also checks the integrity and authenticity of the next level firmware before executing it.

The actual functions of the secure boot depend on the availability of TrustZone features, and on the firmware stored in the device. However, the secure boot typically initializes the secure storage, and installs on-the-fly decryption keys in the OTFDEC to be able to use encrypted firmware stored in an external flash memory.

The device Trusted Firmware-M (TFM) application, supported by the STM32 ecosystem, provides a root of trust solution including secure boot functions.

In the devices, the secure boot takes the benefit of hardware security features such as:

- Resource isolation using TrustZone (see *Section 4.4*)
- Temporal isolation using hide protection (see *Section 4.5.1*)
- Secure execution (see *Section 4.6*)
- Secure update (see *Section 4.3*)
- Secure storage, with associated cryptographic engines if available (see *Section 4.7* and *Section 4.8*)

4.2.1 Unique boot entry and BOOT_LOCK

When TrustZone is activated (TZEN = 1) and BOOT_LOCK secure option bit is cleared, the application selects a boot entry point located either in the system flash memory (see *Section 4.2.2*), or in the secure user flash memory, at the address defined by SECBOOTADD0 option bytes.

When TrustZone is activated (TZEN = 1) and BOOT_LOCK secure option bit is set, the device unique boot entry is the unmodifiable secure address defined by SECBOOTADD0 option bytes. All these option bytes cannot be modified by the application anymore when BOOT_LOCK is set.

Note: As long as it is cleared, the BOOT_LOCK option bit can be set without any constraint. But once set, the BOOT_LOCK option bit cannot be cleared when RDP level > 0.

For more information on the boot mechanisms, refer to *Section 3: Boot modes*.

4.2.2 Immutable root of trust in system flash memory

The STMicroelectronics immutable code includes secure runtime services that can be called at runtime, when a secure application sets the SYSCFG_RSSCMDR register to a nonnull value, before triggering a system reset. This runtime feature is deactivated when the BOOT_LOCK secure option bit is set.

4.3 Secure update

The secure firmware update is a secure service that runs after a secure boot. Its actual functions depend on the availability of the TrustZone features and on the firmware stored in the device.

The device Trusted Firmware allows the update of the microcontroller program with new firmware versions, adding new features and correcting potential issues. The update process is performed in a secure way to prevent unauthorized updates and access to confidential on-device data.

A firmware update can be done either on a single firmware image including both secure and nonsecure parts, or on the secure (respectively nonsecure) part of the firmware image, independently.

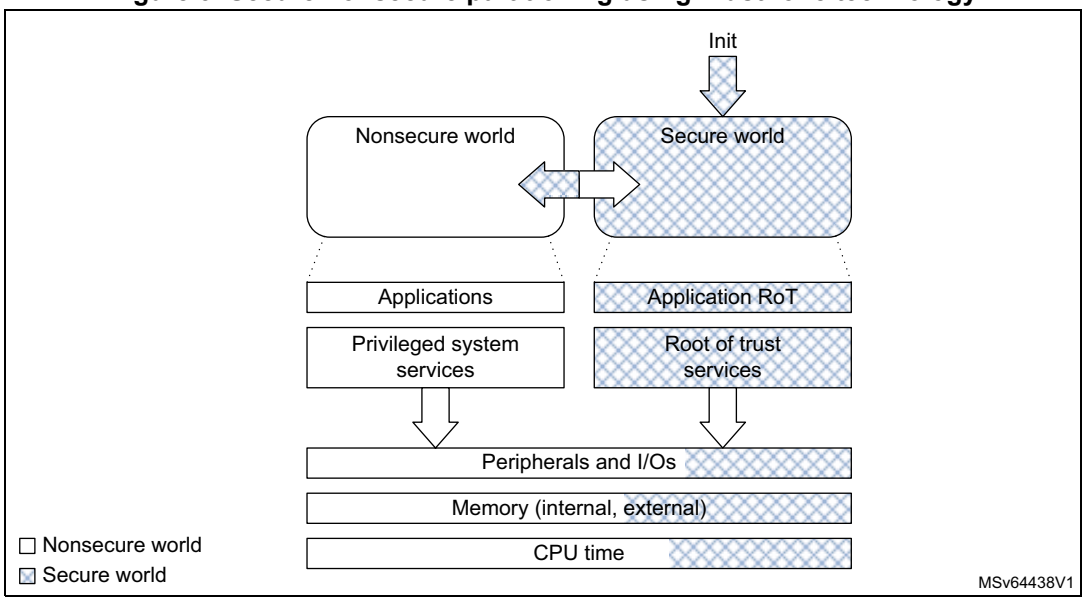
In the devices, the secure update application leverages the hardware security.

- Secure boot (see *Section 4.2*)
- Resource isolation using TrustZone (see *Section 4.4*)
- Temporal isolation using hide protection (see *Section 4.5.1*)
- Secure execution (see *Section 4.6*)
- Secure storage, with associated cryptographic engines (see *Section 4.7* and *Section 4.8*)

4.4 Resource isolation using TrustZone

In the devices, the hardware and software resources can be partitioned so that they exist either in the secure world or in the nonsecure world, as shown in *Figure 3*.

Figure 3. Secure/nonsecure partitioning using TrustZone technology



Note: The initial partitioning of the platform is under the responsibility of the secure firmware executed after reset of the device.

Thanks to this resource isolation technology, the secure world can be used to protect critical code against intentional or unintentional tampering from the more exposed code running in the nonsecure world.

Note: The secure code is typically small and rarely modified, while nonsecure code is more exposed, and prone to firmware updates.

4.4.1 TrustZone security architecture

The Armv8-M TrustZone technology is a comprehensive hardware architecture that proposes to developers a comprehensive, holistic protection across the entire processor and system. The device TrustZone hardware features include:

- The Armv8-M mainline security extension of Cortex-M33, enabling a new processor secure state, with its associated secure interrupts
- The dynamic allocation of memory and peripherals to TrustZone using eight security attribution unit (SAU) regions of Cortex-M33
- A global TrustZone framework (GTZC), extending the TrustZone protection against transactions coming from other masters in the system than the Cortex-M33
- TrustZone-aware embedded flash memory and peripherals

Note: The TrustZone security is activated by the TZEN option bit in the FLASH_OPTR register.

4.4.2 Armv8-M security extension of Cortex-M33

The Arm security extension of the Cortex-M33 is an evolution, not a revolution. It uses the programmer model from earlier Cortex-M subfamilies like Cortex-M4. Indeed, Armv8-M is architecturally similar to Armv7-M, using the same 32-bit architecture, the same memory mapped resources protected with an MPU. Armv8-M also uses the nested vectored interrupt controller (NVIC).

The Armv8-M TrustZone implementation is composed of the following features:

- A new processor state, with almost no additional code/cycle overhead, as opposed to Armv8-A TrustZone, that uses a dedicated exception routine for triggering a secure/nonsecure world change
- Two memory map views of a shared 4-Gbyte address space
- A low interrupt latency for both secure and nonsecure domains, and a new interrupt configuration for security grouping and priority setting
- Separated exception vector tables for the secure and nonsecure exceptions
- Micro-coded context preservation
- banking of specific registers across secure/nonsecure states, including stack pointers with stack-limit checkers
- Banking of the following Cortex-M33 programmable components (two separate units for secure and nonsecure):
  - SysTick timer
  - MPU configuration registers (eight MPU regions in secure, eight in nonsecure)
  - Some of the system control block (SCB) registers
- New system exception (SecureFault) for handling of security violations
- Configurable debug support, as defined in *Section 4.10*

For more information, refer to the Cortex-M33 programming manual (PM0264).

4.4.3 Memory and peripheral allocation using IDAU/SAU

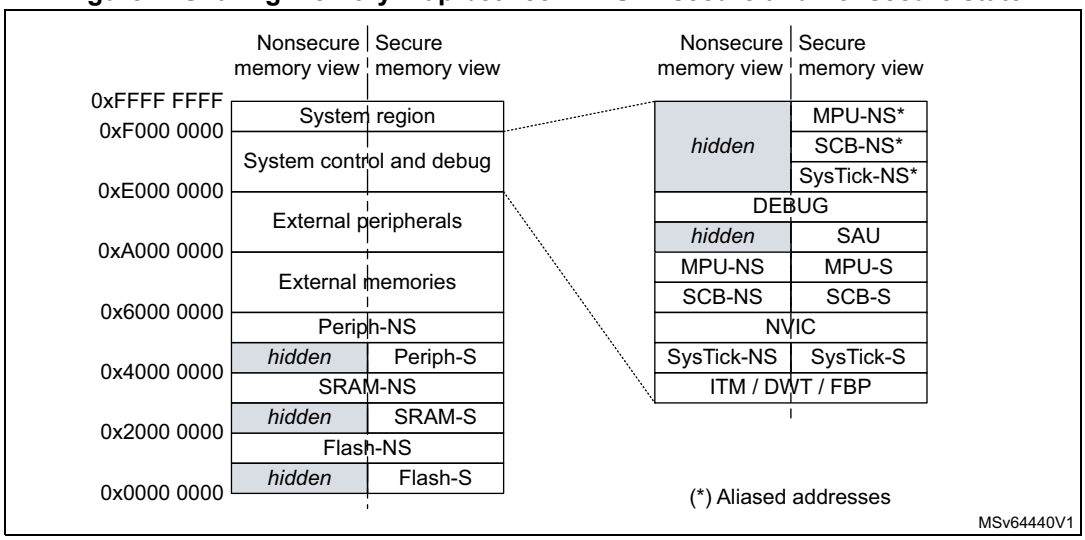
Security attributes

The Armv8-M nonsecure memory view (see *Figure 4*) is similar to Armv7-M (that can be found in Cortex-M4), with the difference that the secure

memory view shows the flash memory, SRAM, and peripherals that are only accessible while the Cortex processor executes in secure state.

Note: Figure 4 shows the 32-bit address space viewed by the secure code after the SAU configuration.

Figure 4. Sharing memory map between CPU in secure and nonsecure state



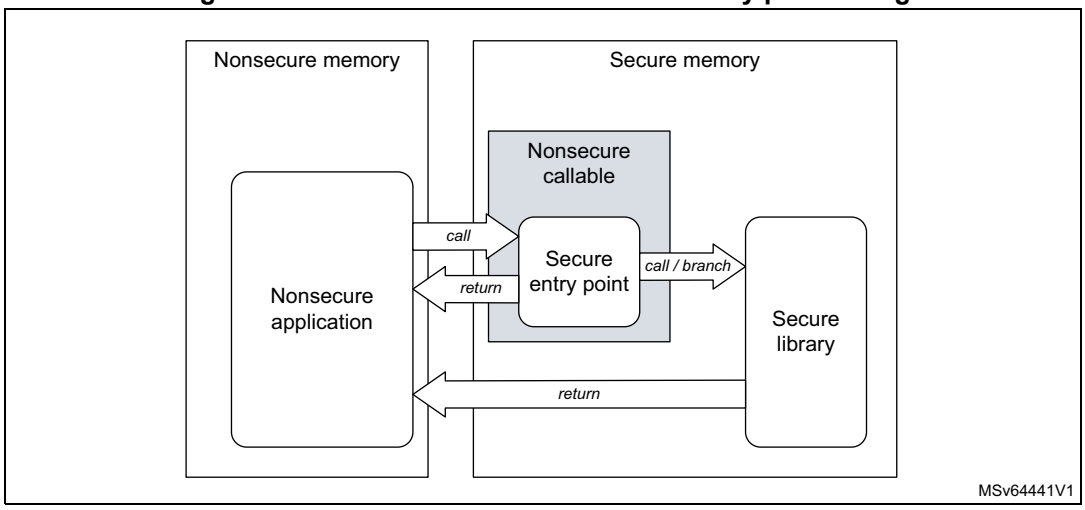
Note: The devices have no support for external memories and external peripherals.

The Cortex processor state (and associated rights) depends on the security attribute assigned to the memory region where it is executed:

- A processor in a nonsecure state only executes from nonsecure (NS) program memory, while a processor in a secure state only executes from secure (S) program memory.
- While running in secure state, the processor can access data from both S and NS memories. Running in nonsecure state, the CPU is limited to nonsecure memories.

To manage transitions to the secure world, developers must create nonsecure callable (NSC) regions that contain valid entry points to the secure libraries. The first instruction in these entry points must be the new **`secure gate`** (SG) instruction, used by the nonsecure code to call a secure function (see *Figure 5*).

Figure 5. Secure world transition and memory partitioning



Programming security attributes

In Cortex-M33, the static implementation-defined attribution unit (IDAU) works in conjunction with the programmable security attribution unit (SAU) to assign a specific security attribute (S, NS, or NSC) to a specific address, as shown in *Table 9*.

Table 9. Configuring security attributes with IDAU and SAU

IDAU security attribution	SAU security attribution(1)	Final security attribution
Nonsecure	Secure	Secure
	Secure-NSC	Secure-NSC
	Nonsecure	Nonsecure
Secure-NSC	Secure	Secure
	Nonsecure	Secure-NSC

1. Defined regions are aligned to 32-byte boundaries.

The SAU can be configured by the Cortex-M33 only in the secure-privileged state. When the TrustZone is enabled, the SAU defaults all addresses as secure (S). A secure boot application can then program the SAU to create NSC or NS regions, as shown in *Table 9*.

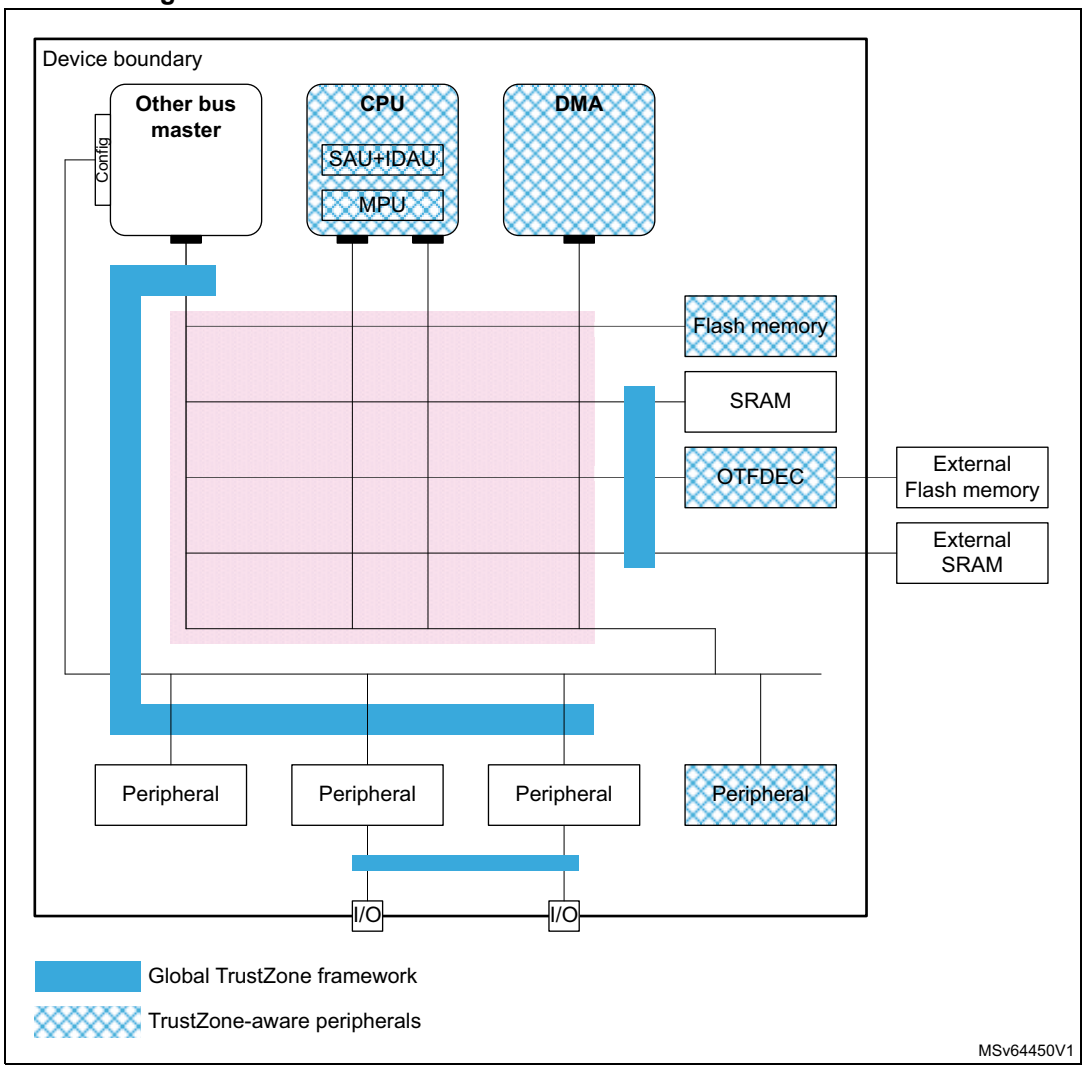
Note: The SAU/IDAU settings are applicable only to the Cortex-M33*. The other masters (like DMA) are not affected by these policies.*

4.4.4 Memory and peripheral allocation using GTZC

Global TrustZone framework architecture

On top of the Armv8-M TrustZone security extension in Cortex-M33, the devices embed complementary security features that reinforce, in a flexible way, the isolation between the secure and the nonsecure worlds. Unlike the SAU/IDAU, the GTZC can protect legacy memories and peripherals against nonsecure transactions coming from other masters than the Cortex-M33.

Figure 6. Global TrustZone framework and TrustZone awareness



Securing peripherals with TZSC

When the TrustZone security is active, a peripheral is either securable through the TZSC block in GTZC, or is natively TrustZone-aware, as shown in the previous figure:

- A securable peripheral or memory is protected by an AHB/APB firewall gate, controlled by the TrustZone security controller (TZSC).
- A TrustZone-aware peripheral or memory is connected directly to AHB or APB interconnect, implementing a specific TrustZone behavior, such as a subset of secure registers or a secure memory area.

TrustZone-aware AHB masters like Cortex-M33 or DMAs, drive a secure signal in the AHB interconnect, according to their security mode, independently of the TZSC.

Note: Like with TrustZone, a peripheral can be made privileged-only with TZSC (see Section 4.5.3). In this case, if this peripheral is master on the interconnect, it automatically issues privileged transactions.

Securing memories with MPCWM and MPCBB

The TZSC provides the capability to manage the security and privilege for securable external memories, programming the MPCWM resource.

Table 10. MPCWM resources

Memory	MPC resource	on-the-fly decryption
XSPI1	GTZC1_MPCWM1	OTFDEC1

The MPCBB blocks in GTZC provide the capability to configure the security and privilege of embedded SRAM blocks, as defined in *Table 11*.

Table 11. MPCBB resources

Memory	MPC resource
SRAM1	GTZC1_MPCBB1
SRAM2	GTZC1_MPCBB2

- Error management (secure-only)
  - Any nonsecure transaction trying to access a secure memory area is considered as illegal. An illegal access event is generated for illegal nonsecure accesses.
  - Any secure transaction trying to access a nonsecure memory area is considered as illegal. An illegal access event is generated for illegal secure accesses.
  - A secure read and write transaction to an MPCBB nonsecure area can be allowed by setting MPCBB SRWILADIS. Instruction fetch remains illegal.
  - The addressed memory area, return all zero data for illegal read and ignoring any illegal write, and an illegal access event is generated. For an illegal instruction access a bus error is generated.

For more information, refer to *Section 5: Global TrustZone® controller (GTZC)*.

Applying GTZC configurations

The TZSC, MPCWM watermarks, and MPCBB blocks can be used in one of the following ways:

- Statically programmed during the secure boot, locked and not changed afterwards
- Dynamically reprogrammed using a specific application code or real-time kernel

When the dynamic option is selected and the configuration is not locked:

- MCPWM nonsecure region size and MPCBB secure blocks region size can be changed by a secure software. This software must be privileged for MPCWM and can be unprivileged if the particular MPCBB block is not privileged-only.
- The secure (respectively privilege) state of each peripheral can be changed writing to GTZC_TZSC_SECCFRGx (respectively GTZC1_TZSC_PRIVCFGRx) registers.

Securing peripherals with TZSC

The TZSC block in GTZC provides the capability to manage the security and the privilege for all securable peripherals. The list of these peripherals can be found in *Section 5: Global TrustZone® controller (GTZC)*.

Note: When the TrustZone is deactivated, the resource isolation hardware GTZC can still be used to isolate peripherals to privileged code only (see Section 4.5.3).

When the TrustZone is activated, peripherals are set as nonsecure and unprivileged after reset.

TrustZone-aware peripherals

The devices include the following TrustZone-aware peripherals:

- GPIOA to GPIOC and GPIOH
- GTZC1_MPCBBx, GTZC1_TZIC and GTZC1_TZSC (MPCWMx)
- OTFDEC1
- EXTI
- Flash memory
- RCC and PWR
- LPDMA
- SYSCFG
- RTC and TAMP
- MCU debug unit DBGMCU
- ICACHE

The way illegal accesses to those peripherals are monitored through the TZIC registers is described in *Section 5: Global TrustZone® controller (GTZC)*. For more details, refer to *Section 4.4.5*.

Illegal access controller with TZIC

The TZIC block in GTZC gathers all illegal access events originated from sources either protected by GTZC or TrustZone-aware peripherals, generating one global secure interrupt towards the NVIC.

TZIC is available only when the system is TrustZone enabled (TZEN = 1). All accesses to TZIC registers must be secured and privileged.

For each illegal event source, a status flag and a clear bit exist. Each illegal event can be masked, not generating an interrupt to the NVIC.

Note: By default, all events are masked.

4.4.5 Managing security in TrustZone-aware peripherals

This section gives more details on how the security is implemented in the TrustZone-aware peripherals listed in the previous section.

Embedded flash memory

When the TrustZone security is enabled through option bytes (TZEN = 1), the whole flash memory is secure after reset and the following protections, shown in *Figure 7*, are available to the application:

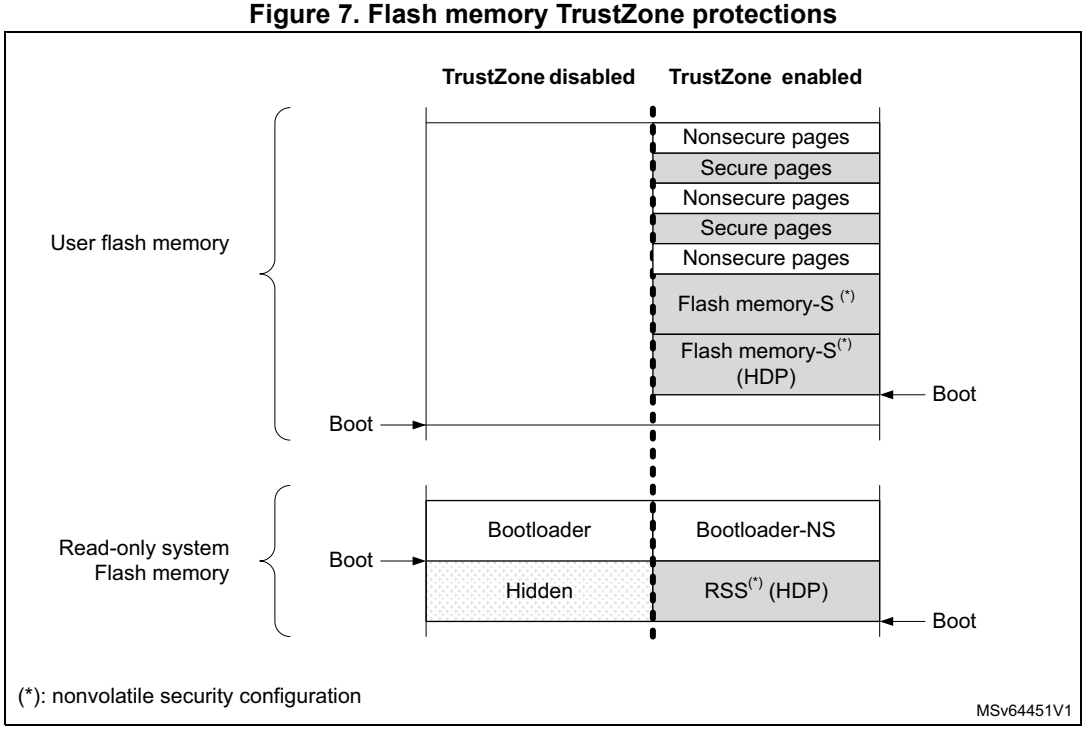
- Non-volatile user secure areas, defined with nonvolatile secure user option bytes
  - Watermark-based secure only area
  - Secure hide protection (HDP) area, stickily hidden after boot
- Volatile user secure pages, defined with volatile secure registers (lost after reset)
  - Any page set as nonsecure (example: outside watermark-based secure only area), can be set as secure on-the-fly using the block-based configuration registers.

Note: All areas are aligned on the flash memory page granularity.

The flash memory area can be configured as secure while it is tagged as nonsecure in Cortex-M33 IDAU/SAU. In this case, nonsecure accesses by the CPU to the flash memory are denied.

Erase or program operations can be available to secure (resp. nonsecure) code only for secure (resp. nonsecure) pages or memory. A flash memory is considered secure if at least one page is secure.

Figure 7. Flash memory TrustZone protections



As shown above, when TrustZone is activated (TZEN = 1), the application code can use the HDP area that is part of the flash watermark-based secure area. Indeed, when the

application sets the HDP_ACCDIS bit, data read, write, and instruction fetch on this HDP area are denied until the next system reset.

For example, the software code in the secure flash HDP area can be executed only once, with any further access to this area denied until the next system reset. Additionally, any flash memory page belonging to an active HDP area cannot be erased anymore.

When the TrustZone is deactivated (TZEN = 0), the volatile/non-volatile secure area features are deactivated and all secure registers are RAZ/WI.

See *Section 7: Embedded flash memory (FLASH)* for more details.

On-the-fly decryption (OTFDEC)

When TrustZone security is enabled (TZEN = 1), the OTFDEC can only be initialized by secure applications. Each encrypted region, once configured can be write-locked until a next OTFDEC reset.

Note: Application secure or nonsecure can verify the initialization context of each region (including the CRC of the keys).

See Section 4.8.2: On-the-fly decryption engine (OTFDEC) for more details.

Direct memory access controller (LPDMA)

When a DMA channel x is defined as secure (SECx = 1 in LPDMA_SECCFGR), the source and destination transfers can be independently set as secure or nonsecure by a secure application using SSEC and DSEC bits in LPDMA_CxTR1. *Table 12* summarizes the security options available in each DMA channel.

Table 12. DMA channel use (security)(1)

Destination type	Secure DMA channel x (SECx = 1)		Nonsecure DMA channel y (SECy = 0)	
	Secure source	Nonsecure source	Secure source	Nonsecure source
Secure destination	OK	OK(2)	Transfer blocked	
Nonsecure destination	OK(3)	OK(4)	Transfer blocked	OK

1. When a transfer is blocked, it completes, but the corresponding writes are ignored, and reads return zeros. Also an illegal access event to TZIC is automatically triggered by the memory/peripheral used as source or destination.
2. If the source is a memory, the transfer is possible only if SSEC = 0, otherwise the transfer is blocked.
3. If the destination is a memory, the transfer is possible only if DSEC = 0, otherwise the transfer is blocked.
4. The memory-to-memory transfer is possible only if SSEC = 0 and DSEC = 0, otherwise it is blocked.

When a channel is configured as secure:

- Registers allocated to this channel (excluding LPDMA_SECCFGR, LPDMA_PRIVCFGR and LPDMA_RCFGLOCKR) are read as zero. Write are ignored for nonsecure accesses. A secure illegal access event may also be triggered to the TZIC peripheral.
  - Writes to LPDMA_SECCFGR and LPDMA_RCFGLOCKR must be secure. For each bit in LPDMA_PRIVCFGR, write must be secure if the corresponding bit in LPDMA_SECCFGR is set.
- In linked-list mode, the loading of the next linked-list data structure from memory is performed with secure transfers.

- When switching to a nonsecure state, the secure application must abort the channel or wait until the secure channel is completed before doing the switch.

Note: DMA secure channels are not available when TrustZone is deactivated.

When a channel is configured as nonsecure in linked-list mode, the loading of the next linked-list data structure from memory is performed with nonsecure transfers. See Section 16: Low-power direct memory access controller (LPDMA) for more details.

Power control (PWR)

When the TrustZone security is activated (TZEN = 1), the selected PWR registers can be secured through PWR_SECCFGR, protecting the following PWR features:

- Low-power mode setup
- Wake-up (WKUP) pins definition
- Voltage detection
- backup domain control

Other PWR configuration bits become secure:

- When the system clock selection is secure in the RCC: the voltage scaling (VOS) configuration become secure
- When a GPIO is configured as secure: its corresponding bit for standby mode retention configuration becomes secure
- When the 2.4 GHz RADIO is configured as secure: its corresponding control bits become secure

See *Section 11: Power control (PWR)* for details.

Secure clock and reset (RCC)

When the TrustZone security is activated (TZEN = 1) and security is enabled in the RCC, the bits controlling the peripheral clocks and resets become TrustZone-aware:

- If the peripheral is securable and programmed as secure in the TZSC, the peripheral clock and reset bits become secure.
- If the peripheral is TrustZone-aware, the peripheral clock and reset bits become secure as soon as at least one function is configured as secure inside the peripheral.

When a peripheral is defined as secure, its related clock, reset, and clock source selection in the RCC, become secure and in some case the selection of clock source during low-power modes as well.

Note: Refer to Table 3 and Table 4 for the list of securable and TrustZone-aware peripherals.

Additionally, the following configurations can be made secure-only using RCC_SECCFGR:

- External clock (such as HSE32 or LSE), internal oscillator (such as HSI16, or LSI)
- PLL
- AHB and APB prescalers
- System clock source, SysTick, and MCO clock output selection
- Reset flag clearing

See *Section 12: Reset and clock control (RCC)* for details.

Real time clock (RTC)

Like all TrustZone-aware peripherals, a nonsecure read/write access to a secured RTC register is RAZ/WI. It also generates an illegal access event that triggers a secure illegal access interrupt if the RTC illegal access event is enabled in the TZIC.

After a backup domain power-on reset, all RTC registers can be read or written in both secure and nonsecure modes. The secure boot code can then change its security setup, making registers alarm A, alarm B, wake-up timer and timestamp secure or not, using RTC_SECCFGR.

When the SEC bit is set in secure-only RTC_SECCFGR:

- Writing the RTC registers is possible only in secure mode.
- Reading RTC_SECCFGR, RTC_PRIVCFGR, RTC_MISR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER, and RTC_CALR is always possible in secure and nonsecure modes. All the other RTC registers can be read only in secure mode.

When SEC is cleared in secure-only RTC_SECCFGR, it is still possible to restrict access in secure mode to some RTC registers by setting dedicated control bits: INITSEC, CALSEC, TSSEC, WUTSEC, ALRASEC, and ALRBSEC.

Note: The RTC security configuration is not affected by an NRST pin reset.

See *Section 33: Real-time clock (RTC)* for more details.

Tamper and backup registers (TAMP)

Like all TrustZone-aware peripherals, a nonsecure read/write access to a secured TAMP register is RAZ/WI. It also generates an illegal access event that triggers a secure illegal access interrupt if the TAMP illegal access event is enabled in the TZIC.

After a backup domain power-on reset, all TAMP registers can be read or written in both secure and nonsecure modes. The secure boot code can change its security setup, making some registers secure or not as needed, using TAMP_SECCFGR register.

When TAMPSEC is set in TAMP_SECCFGR:

- Writing the TAMP registers is possible only in secure mode. Backup registers have their own write protection (see below).
- Reading the TAMP registers (except for TAMP_SECCFGR, TAMP_PRIVCFGR and TAMP_MISR) returns zero if the access is nonsecure. Backup registers have their own read protection (see below).

The application can also:

- Make TAMP_COUNTR register read and write secure-only by setting the CNT1SEC bit in TAMP_SECCFGR secure register
- In backup registers increase security for two of the three protection zones configured using BKPRWSEC[7:0] and BKPWSEC[7:0] bitfields in TAMP_SECCFGR:
  - Protection zone 1 is read secure, write secure
  - Protection zone 2 is read nonsecure, write secure
  - Protection zone 3 is read nonsecure, write nonsecure

Note: The TAMP security configuration is not affected by an NRST reset.

See *Section 34: Tamper and backup registers (TAMP)* for more details.

General-purpose I/Os (GPIO)

When the TrustZone security is activated (TZEN = 1), each I/O pin of the GPIO port can be individually configured as secure through the GPIOx_SECCFGR registers. Only a secure application can write to GPIOx_SECCFGR registers. After boot, each I/O pin of the GPIO is set as secure.

When an I/O pin is configured as secure, its corresponding configuration bits for alternate function (AF), mode selection (MODE), and I/O data are RAZ/WI in case of nonsecure access.

When digital alternate function is used (input/output mode), to protect the data transiting from/to the I/O managed by a secure peripheral, the devices add a secure alternate function gate on the path between the peripheral and its allocated I/Os:

- If the peripheral is secure, the I/O pin must also be secure to allow input/output of data.
- If the peripheral is not secure, the connection is allowed regardless of the I/O pin state.

The TrustZone-aware logic around GPIO ports, used as an alternate function, is summarized in *Table 13*.

Table 13. Secure alternate function between peripherals and allocated I/Os

Security configuration		Alternate function logic		Comment
Peripheral	**Allocated I/O pin**	**Input**	**Output**	
Secure	Secure	I/O data	Peripheral data	-
Nonsecure				Out of reset configuration
Secure	Nonsecure	0	0	-
Nonsecure		I/O data	Peripheral data	

When an analog function with an analog switch is used, the connection to the peripherals listed in *Table 14*, is blocked by hardware when the peripheral is nonsecure and the I/O is secure.

Table 14. Nonsecure peripheral functions not connected to secure I/Os

Peripheral	Analog function(1)	Input	Output	How to set the peripheral or the function as secure
ADC4	ADC4_INy (y = 1 to 10)	X	-	Set ADC4SEC in GTZC1_TZSC

1. Used to find the I/O corresponding to the signal/function on the package (refer to the product datasheet).

Finally, regarding GPIO and security, *Table 15* summarizes the list of peripheral functions that do not have any hardware protection linked to TrustZone. The listed signals (input and/or outputs) are not blocked when the I/O is set as secure and the associated peripheral function is nonsecure.

For example, when a secure application sets an I/O port pin as secure to be used as LPTIM2_OUT, if the RTC_OUTx is nonsecure, it can be programmed to output data to the I/O port pin, potentially causing malfunction to the secure application.

Similarly, when a secure application sets an I/O port pin as secure to be used as USART1_TX, if TAMP_INx is nonsecure, it can be programmed to capture the USART input traffic.

It is important that, for each case described in *Table 15*, the secure application decides if a potential effect on data integrity or confidentiality is critical or not. For example, if the USART situation described above is not acceptable (data transiting on secure USART is confidential), then the secure application must configure the TAMP as secure, even if it does not use it.

Note: How to make a peripheral secure is summarized in the right column of Table 15.

Table 15. Nonsecure peripheral functions that can be connected to secure I/Os

Peripheral	Signal(1)	Input	Output	How to set the peripheral or the function as secure
TAMP	TAMP_INx (x = 1 to 6)	X	-	Set TAMPSEC in TAMP_SECCFGR
	TAMP_OUTx (x = 1 to 6)	-	X	
RTC	RTC_OUTx (x = 1, 2)	-	X	Set SEC in RTC_SECCFGR
	RTC_TS	X	-	Set TSSEC in RTC_SECCFGR
PWR	WKUPx (x = 1 to 8)	X	-	Set WUPxSEC in PWR_SECCFGR
RCC	LSCO	-	X	Set LSESEC in RCC_SECCFGR
EXTI	EXTIx (x = 0 to 15)	X	-	Set SECx bit in EXTI_SECCFGR

1. To find the I/O corresponding to the signal/function on the package, refer to the product datasheet.

Refer to *Section 13: General-purpose I/Os (GPIO)* for more details.

Extended interrupts and event controller (EXTI)

When the TrustZone security is activated (TZEN = 1), the EXTI is able to protect event register bits from being modified by nonsecure accesses. The protection can individually be activated per input event via the register bits in EXTI_SECCFGR1. When an input event is configured as secure, only a secure application can change the configuration (including security if applicable), change the masking or clear the status of this input event.

The security configuration in EXTI_SECCFGR1 can be globally locked after reset in EXTI_LOCKR.

See *Section 18: Extended interrupts and event controller (EXTI)* for more details.

System configuration controller (SYSCFG)

Like all TrustZone-aware peripherals, when the TrustZone security is activated (TZEN = 1), a nonsecure read/write access to a secured SYSCFG register is RAZ/WI. Such access also generates an illegal access event that triggers a secure illegal access interrupt if the SYSCFG illegal access event is not masked in the TZIC.

See *Section 14: System configuration controller (SYSCFG)* for more details.

Microcontroller debug unit (DBGMCU)

The MCU debug component (DBGMCU) helps the debugger, providing support for:

- Low-power modes behavior during debug
- Peripheral freeze during debug, applicable to I2Cs, IWDG, timers, low-power timers, and DMA channels

The DBGCMU is a TrustZone-aware peripheral, managing accesses to its control registers as described in *Table 16*.

Table 16. TrustZone-aware DBGMCU accesses management

Debug profile	Peripheral status(1)	DBG_xx_STOP control bits	
		Write access	Read access
Nonsecure invasive (SPIDEN = 0)	NS	Yes (S(2) or NS)	Yes (S or NS)
	S	None (S or NS)	
Secure invasive (SPIDEN = 1)	NS	Yes (S or NS)	
	S	Yes (S only)	

1. As reported by the GTZC or the TrustZone-aware peripheral feature.
2. Secure access from the debugger is converted to nonsecure access in the device.

Refer to *Section 41.13: Microcontroller debug unit (DBGMCU)* for more details.

4.4.6 Activating TrustZone security

The TrustZone is deactivated by default in the device. It can be activated by setting the TZEN user option bit in FLASH_OPTR when in RDP Level 0. Once TZEN has changed from 0 to 1, the default security state, after reset, is always the following:

- CPU subsystem
  - Cortex-M33 exits reset in secure state, hence the boot address must point toward a secure memory area.
  - All interrupt sources are secure (in NVIC).
  - The memory mapped viewed by the CPU through IDAU/SAU is fully secure.
- Embedded flash memory
  - Flash memory nonvolatile secure area (with the HDP zone), are defined with nonvolatile registers FLASH_SECWMR. The default secure option bytes setup is all user flash secure, without HDP area defined.
  - Volatile block-based security attributions of the flash memory are nonsecure.The secure boot code can change this setup, making blocks secure.
- Embedded SRAM memories
  - All SRAMs are secure, as defined in GTZC/MPCBB (see *Section 4.4.4*). The secure boot code can change this security setup, making blocks nonsecure.
- External memories
  - All memory devices connected to the XSPI are secure, as defined in GTZC/MPCWM (see *Section 4.4.4: Memory and peripheral allocation using GTZC*). The secure boot code can change this security setup, making the components secure or not.

- All GPIOs are secure.
- All LPDMA channels are nonsecure.
- Backup registers are nonsecure.
- Peripherals and GTZC:
  - Securable peripherals are nonsecure and unprivileged.
  - TrustZone-aware peripherals are nonsecure, with their secure configuration registers being secure.
  - All illegal access interrupts in GTZC1_TZIC are disabled.

Note: Refer to Table 3 and Table 4 for the list of securable and TrustZone-aware peripherals.

4.4.7 Deactivating TrustZone security

Once TrustZone is activated, it can only be deactivated during an RDP regression to Level 0.

Note: Such an RDP regression triggers the erasure of embedded memories (flash memory, SRAM1, SRAM2, PKA SRAM and TAMP backup registers), and the reset of all peripherals, including all crypto engines including OTFDEC.

Note: Before launching an RDP regression, the software must invalidate the ICACHE and wait for the BUSYF bit to get cleared.

After the TrustZone deactivation, most of the features mentioned in Section 4.4 are no longer available:

- The nonvolatile secure area of the embedded flash memory is deactivated, including the HDP area.
- The NVIC only manages nonsecure interrupts.
- All secure registers in TrustZone-aware peripherals are RAZ/WI.

Note: When the TrustZone is deactivated, the resource isolation using privilege stays available (see Section 4.5.3 for details).

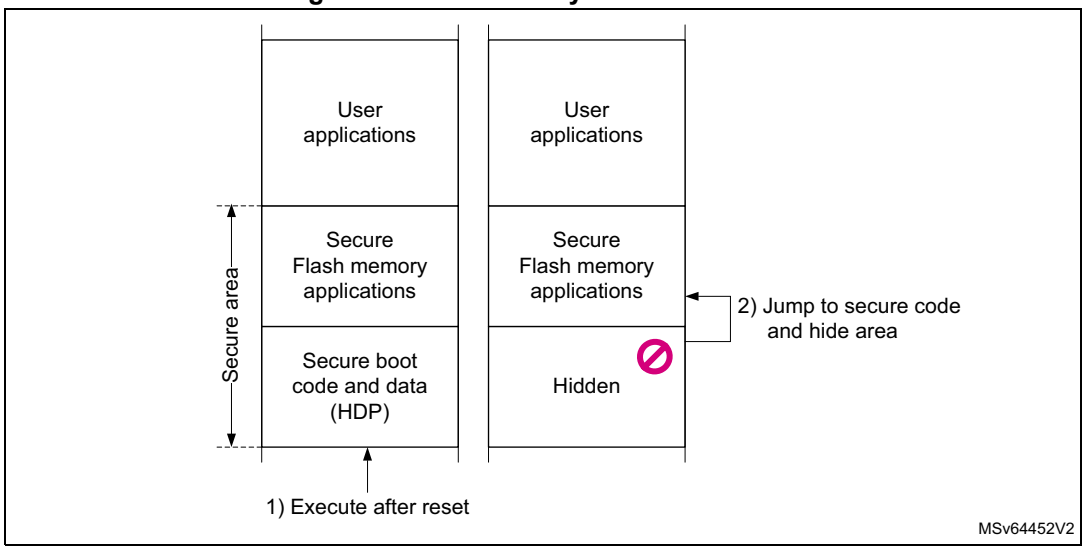
4.5 Other resource isolations

These are hardware mechanisms offering an additional level of isolation on top of the TrustZone technology.

4.5.1 Temporal isolation using secure hide protection (HDP)

When the TrustZone security is enabled (TZEN = 1), the embedded flash memory allows an HDP area in the watermarked-secure area (8-Kbyte page granularity) to be defined. The code executed in this HDP area (with its related data and keys) can be hidden after boot until the next system reset. The hide protection principle is shown in *Figure 8*.

Figure 8. Flash memory secure HDP area



When the HDPEN and HDP_ACCDIS bit is set, data read, write, and instruction fetch on the area defined by SECWM_PSTRT and HDP_PEND option bytes, are denied until the next device reset. The end of the HDP area can be dynamically extended by the HDP application using the HDP_PEXT.

Note: Mass erase aborts when it contains a write-protected area (WRP or HDP area).

The HDP area can be resized by a secure application if the area is not hidden and if RDP Level ≠ 2.

4.5.2 RSSLIB functions

The RSS provides runtime services thanks to RSS library. As other microcontroller peripherals features and mapping, the RSS library functions are exposed to user within the CMSIS device header file provided by the STM32Cube firmware package. Please refer to UM3131 to get more details regarding STM32Cube firmware package. RSS library functions are named RSSLIB functions hereafter.

The user firmware calls RSSLIB functions using RSSLIB_PFUNC C defined macro, that points to a location within nonsecure system memory. Hence prior calling RSSLIB functions, the secure user firmware must define a nonsecure region above this location within SAU of the Cortex®-M33. This nonsecure region starts from RSSLIB_SYS_FLASH_NS_PFUNC_START up to RSSLIB_SYS_FLASH_NS_PFUNC_END. These last addresses are provided within the CMSIS device header file. The user can set this nonsecure region either by using the CMSIS system partition header file or by implementing its own code for SAU setup. The CMSIS system partition header file is part of the STM32Cube firmware package.

RSSLIB functions are split between nonsecure callable and secure callable function.

The RSS library functions are described within sections hereafter.

CloseExitHDP

Bootloader ID:

CloseExitHDP1 function

Secure attribute:

Secure callable function.

Prototype:

```
uint32_t CloseExitHDP(uint32_t HdpArea, uint32_t
VectorTableAddr)
```

Arguments:

- HdpArea:
  Input parameter: RSSLIB_HDP_AREA1_Msk.
- VectorTableAddr:
  Input parameter, address of the next vector table to apply.
  The vector table format is the one used by the Cortex®-M33 core.

Description:

The user calls CloseExitHDP to close flash HDP secure memory area and jump to the reset handler embedded within the vector table which address is passed as input parameter.

CloseExitHDP sets the SP provided by the passed vector table, however it is up to the caller to first set the new vector table. Then it clears all general-purpose.

A Cortex®-M33 registers (r0, r1, …) before jumping to new vector table reset handler.

On successful execution, the function does not return and does not push LR onto the stack.

In case of failure (bad input parameter value), this function returns RSSLIB_ERROR.

Refer to section *Section 7.5.3: Secure hide protection (HDP)* to get more details on flash memory HDP protection.

4.5.3 Resource isolation using Cortex privileged mode

In parallel to the TrustZone isolation described in *Section 4.4*, the hardware and software resources can be partitioned so that they are restricted to software running in Cortex privileged mode.

Thanks to this hardware isolation technology, available even if TrustZone is deactivated (TZEN = 0), critical code or data can be protected against intentional or unintentional tampering from the more exposed unprivileged code.

Memory and peripheral privileged allocation using MPU

The Cortex-M33 MPU divides the unified memory into eight regions, each aligned to a multiple of 32 bytes. Each memory regions can be programmed to generate faults when accessed inappropriately by unprivileged software.

Memory and peripheral privileged allocation using GTZC

For the Cortex-M33 master, to complement the coarse isolation provided by the MPU, the GTZC reinforces, in a flexible way, the isolation between privileged and unprivileged tasks, for peripherals and selected memories.

- Securing peripherals with TZSC (privileged-only)

  In the devices, a peripheral is either securable privileged-only through GTZC, or is natively privileged-aware:

  – A securable privileged-only peripheral or memory is protected by an AHB/APB firewall gate that is controlled by the TZSC.

  – A privileged-aware peripheral or memory is connected directly to AHB or APB interconnect, implementing a specific behavior such as a subset of registers or a memory area is privileged-only.

  The list of securable peripherals can be found in *Table 3*.

- Securing internal memories with MPCBB (privileged-only)

  The GTZC provides the capability to configure the privilege level of embedded SRAM blocks, programming the MPCBB resources defined in *Section 4.4.4: Memory and peripheral allocation using GTZC*.

- Securing external memories with MPCWM (privileged-only)

  The GTZC provides the capability to configure the privilege level of external memories, programming the MPCWM resources defined in *Section 4.4.4: Memory and peripheral allocation using GTZC*.

- Error management (privileged-only)

  – Any unprivileged transaction trying to access a privileged resource is considered as illegal. There is no illegal access event generated for illegal unprivileged read and write accesses.

  – The addressed resource follows a silent-fail behavior, returning all zero data for read and ignoring any write.

  – When an illegal unprivileged access occurs, no bus error is generated, except when this access is an instruction fetch, accessing a privileged memory or a peripheral register.

Managing security in privileged-aware peripherals

TrustZone-aware peripherals also implement privileged-only access mode. The privileged protection is valid even if TZEN = 0:

- *Embedded flash memory*

  By default all embedded flash registers can be read or programmed in both privileged and unprivileged modes.

  When secure privileged bit SPRIV is set in FLASH_PRIVCFGR, reading and writing the flash secure registers are possible only in privileged mode. Write access to this bit is ignored if TrustZone is deactivated (TZEN = 0).

  When the nonsecure privileged bit NSPRIV is set in FLASH_PRIVCFGR, reading and writing the flash nonsecure registers are possible only in privileged mode.

  Regarding privileged protection of the embedded flash memory, the devices offer the following features:

  – The system flash memory can be accessed in both privileged and unprivileged modes.
  – Each watermark-based secure area, including its secure HDP area, is accessible in secure-privileged and secure-unprivileged mode, if applicable.
  – Each 8-Kbyte page of the embedded flash memory can be programmed on-the-fly as privileged only, using the block-based privileged configuration register FLASH_PRIVBBRx. An unprivileged page is accessible either by a privileged or by an unprivileged access.

Note: Switching a page from privileged to unprivileged does not erase the content of the page.

When applicable, an erase or program operation is always available to privileged code, and is available to unprivileged code only for unprivileged pages or unprivileged memory.

- On-the-fly decryption (OTFDEC)

  When privileged (PRIV = 1 in OTFDEC_PRIVCFGR) the OTFDEC can only be initialized and read by a privileged application.

- Direct memory access controllers (LPDMA)

  When a DMA channel x is defined as privileged (PRIVx = 1 in LPDMA_PRIVCFGR), special rules apply when accessing privileged/unprivileged source or destination. Those rules are summarized in *Table 17*).

Table 17. DMA channel use (privilege)

Destination	Privileged DMA channel x (PRIVx = 1)		Unprivileged DMA channel y (PRIVy = 0)	
	Privileged source	Unprivileged source	Privileged source	Unprivileged source
Privileged	OK		Transfer blocked(1)	
Unprivileged			Transfer blocked	OK

1. When a transfer is blocked, the transfer completes but the corresponding writes are ignored, and reads return zeros.

See *Section 16: Low-power direct memory access controller (LPDMA)* for more details.

- *Power control (PWR)*

  By default, after a reset, all PWR registers but PWR_PRIVCFGR, can be read or written in both privileged and unprivileged modes.

  When secure privileged bit SPRIV is set in PWR_PRIVCFGR, reading and writing the PWR securable registers are possible only in privileged mode. Write access to this bit is ignored if TrustZone is disabled (TZEN = 0).

  When the nonsecure privileged bit NSPRIV is set in PWR_PRIVCFGR, reading and writing the PWR nonsecure registers are possible only in privileged mode.

  See *Section 11: Power control (PWR)* for details.

- *Secure clock and reset (RCC)*

  By default, after a reset, all RCC registers but RCC_PRIVCFGR can be read or written in both privileged and unprivileged modes.

  When the secure privileged bit SPRIV is set in RCC_PRIVCFGR, reading and writing the RCC securable bits are possible only in privileged mode. Write access to this bit is ignored if TrustZone is disabled (TZEN = 0).

  When the nonsecure privileged bit NSPRIV is set in RCC_PRIVCFGR, reading and writing the RCC nonsecure bits are possible only in privileged mode.

  See *Section 12: Reset and clock control (RCC)* for details.

- *Real time clock (RTC)*

  By default after a backup domain reset, all RTC registers but RTC_PRIVCFGR, can be read or written in both privileged and unprivileged modes.

  When PRIV bit is set in privileged-only RTC_PRIVCFGR:

  – Writing the RTC registers is possible only in privileged mode.
  – Reading the RTC_SECCFGR, RTC_PRIVCFGR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in privileged and unprivileged modes.

  All the other RTC registers can be read only in privileged mode.

  When the PRIV bit is cleared in the privileged-only RTC_PRIVCFGR register, it is still possible to restrict access to privileged mode to some RTC registers by setting dedicated control bits: INITPRIV, CALPRIV, TSPRIV, WUTPRIV, ALRAPRV, or ALRBPRIV.

  See *Section 33: Real-time clock (RTC)* for details.

- *Tamper and backup registers (TAMP)*

  By default after a backup domain reset, all TAMP registers but TAMP_PRIVCFGR can be read or written in both privileged and unprivileged modes.

  When PRIV bit is set in privileged-only TAMP_PRIVCFGR:

  – Writing the TAMP registers is possible only in privileged mode, except for the backup registers and the monotonic counters that have their own protection setting.
  – Reading the TAMP_SECCFGR or TAMP_PRIVCFGR is always possible in privileged and unprivileged modes. All the other TAMP registers can be read only in privileged mode, except for the backup registers and the monotonic counters that have their own protection setting.

The application can also:

- Make TAMP_COUNT1R register read and write privileged-only by setting the CNTPRIV bit in TAMP_PRIVCFGR
- Increase security for two of the three protection zones in backup registers, using BKPRWPRIV and BKPWPRIV bits in TAMP_PRIVCFGR:
  - Make protection zone 1 read privileged, write privileged.
  - Make protection zone 2 read privileged or unprivileged, write privileged.
  - Protection zone 3 is always read and write privileged or unprivileged.

- *General-purpose I/Os (GPIO)*

  All GPIO registers can be read and written by privileged and unprivileged accesses, whatever the security state (secure or nonsecure).

- *Extended interrupts and event controller (EXTI)*

  The EXTI peripheral can protect event register bits from being modified by unprivileged accesses. The protection is individually activated per input event via the register bits in the privileged-only EXTI_PRIVCFGR1 register. When an input event is configured as privileged, only a privileged application can change the configuration (including security if applicable), change the masking or clear the status of this input event.

  The security configuration in EXTI_PRIVCFGR1 can be globally locked after reset in EXTI_LOCKR.

  See *Section 18: Extended interrupts and event controller (EXTI)* for more details.

- *System configuration controller (SYSCFG)*

  All SYSCFG registers can be read and written in both privileged and unprivileged modes, except:

  - FPUSEC bit in SYSCFG_SECCFGR registers (privileged only)
  - SYSCFG registers for CPU configuration: SYSCFG_CSLCKR, SYSCFG_FPUIMR, and SYSCFG_CNSLCKR

  See *Section 14: System configuration controller (SYSCFG)* for more details.

4.6 Secure execution

Through a mix of special software and hardware features, the devices ensure the correct operation of their functions against abnormal situations caused by programmer errors, software attacks through network access, or local attempt for tampering code execution.

This section describes the hardware features specifically designed for secure execution.

4.6.1 Memory protection unit (MPU)

The Cortex-M33 includes a memory protection unit (MPU) that can restrict the read and write accesses to memory regions (including regions mapped to peripherals), based on one or more of the following parameters

- Cortex-M33 operating mode (privileged, unprivileged)
- data/instruction fetch

The memory map and the programming of the nonsecure and secure MPUs split memory into regions (up to eight per MPU). Secure MPU is available only when TrustZone is activated.

4.6.2 Embedded flash memory write protection

The embedded flash memory write protection (WRP) prevents illegal or unwanted write/erase to special sections of the embedded flash memory user area (system area is permanently write protected).

Write protected area is defined through the option bytes, writing the start and end addresses: two write-protected areas can be defined, with the granularity of flash page size.

WRP areas can be modified through option byte changes unless the corresponding FLASH_WRPA/BR has its UNLOCK option bit cleared (meaning ROM emulation). UNLOCK can be set only when regressing from RDP Level 1 to Level 0.

Note: Mass erase aborts when it contains a write-protected area (WRP or HDP area).

4.6.3 Tamper detection and response

Principle

The devices include active protection of critical security assets attacks, with the following features:

- Erasing of device secrets upon tamper detection
- Improved guarantee of safe execution for the CPU and its associated security peripherals, including:
  - Out-of-range clocking LSE detection
  - Security watchdog IWDG clocked by the internal oscillator LSI
  - Possible selection of internal oscillator HSI16 as system clock
  - Secure ADC4 window watchdog monitoring

See *Section 34: Tamper and backup registers (TAMP)* for more details.

Tamper detection sources

The devices support six active input/output pins, allowing three independent active-tamper meshes, or up to five meshes if the same output pin is shared by several input pins (for a total of six active-tamper I/Os).

The active pins when clocked by the LSE are functional in all system operating modes (Run, Sleep, Stop, and Standby).

Detection time is programmable, and a digital filtering is available (tamper triggered after two false comparisons in four consecutive comparison samples).

Note: Timestamps are automatically generated when a tamper event occurs.

The internal tamper sources are listed in *Table 18*.

Table 18. Internal tampers in TAMP

Tamper input	Tamper source
itamp3	LSE clock security monitoring
itamp5	RTC calendar overflow (rtc_calovf)
itamp6	JTAG/SWD access when RDP > 0

Table 18. Internal tampers in TAMP (continued)

Tamper input	Tamper source
itamp7, 12, 13	Voltage monitoring (V_{CORE}, V_{SENSE}) through ADC4 analog watchdog, functional down to Stop 1 mode
itamp8	TAMP monotonic counter overflow
itamp9	Fault generation for cryptographic peripherals (PKA, AES, RNG)
itamp11	IWDG timeout and potential tamper (IWDG reset when at least one enabled tamper flag is set)

Response to tampers

Each source of tamper in the device can be configured to trigger the following events:

- Generate an interrupt, capable of waking up the device from Stop 0, 1, 2, and Standby modes, and only for passive tampers, from Stop 3 mode (see TAMPxMSK bits in TAMP_CR2 register).
- Generate a hardware trigger for the low-power timers.
- Erase device secrets if the corresponding TAMPxNOER bit in TAMP_CR2 (for tamper pins) or ITAMPxNOER bit in TAMP_CR3 (for internal tamper) is cleared. The erasable secrets are:
  - Keys stored in AES, HASH, and OTFDEC (encrypted flash memory regions are read as zero)
  - Keys stored in PKA SRAM
  - Secrets stored in SRAM2 and CPU instruction cache memory.

Read/write accesses by software to all these secrets can be blocked, by setting the BKBLOCK bit in TAMP_CR2. The device secrets access is possible only when BKBLOCK is cleared, and no tamper flag is set for any enabled tamper source.

Device secrets erase is also triggered by setting the BKERASE bit in TAMP_CR2.

Software filtering mechanism

Each tamper source can be configured not to launch an immediate erase, by setting the corresponding TAMPxPOM bit in TAMP_CR2 (for external tamper pin) or ITAMPxPOM bit in TAMP_CR3 (for internal tamper).

In such a situation, when the tamper flag is raised, access to below secrets is blocked until all tamper flags are cleared:

- Backup registers: read-as-zero, write-ignored
- SRAM2: read-as-zero, write-ignored
- AES, and HASH peripherals: automatically reset.
- PKA peripheral: reset, with memory use blocked (meaning PKA not usable)

Once the application, notified by the tamper event, analyzes the situation, there are two possible cases:

- The application launches secrets erase with a software command (confirmed tamper).
- The application just clears the flags to release secrets blocking (false tamper).

Note: If the tamper software fails to react to such a tamper flag, an IWDG reset triggers automatically the erasure of secrets.

Details on tamper protection are found in *Section 4.6.3*, while TAMP TrustZone features are briefly described in *Section 4.4.5*.

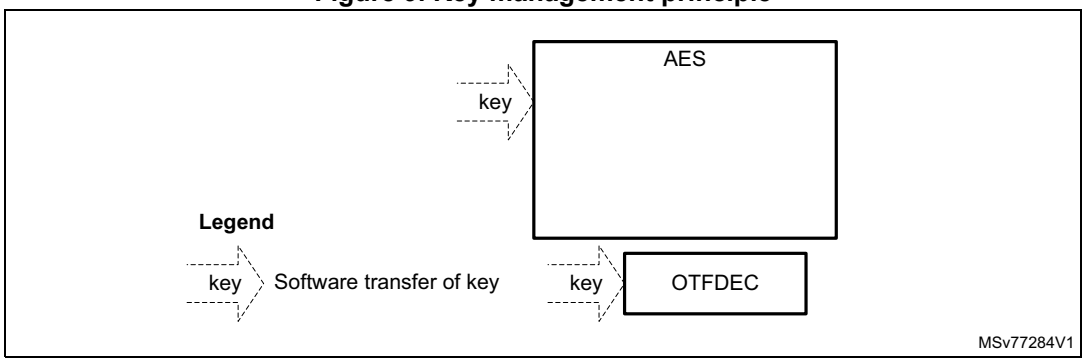
4.7 Secure storage

A critical feature of any security system is how long-term keys are stored, protected, and provisioned. Such keys are typically used to load a boot image, or to handle critical user data.

Figure 9 shows how key management service application can use the AES engine, for example to compute external image decryption keys. A nonvolatile key can be stored in the embedded secure HDP area (see *Section 4.5.1*).

Figure 9 also shows keys that are manipulated by software.

Figure 9. Key management principle



4.7.1 Unique ID

The devices store a 96-bit ID that is unique to each device (see *Section 42: Device electronic signature (DESIG)*).

Application services can use this unique identity key to identify the product in the cloud network, or to make it difficult for counterfeit devices or clones to inject untrusted data into the network.

4.8 Crypto engines

The devices implement state-of-the-art cryptographic algorithms featuring key sizes and computing protection as recommended by national security agencies such as NIST for the U.S.A, BSI for Germany or ANSSI for France. Those algorithms are used to support privacy, authentication, integrity, entropy, and identity attestation.

The crypto engine embedded in STM32 reduces weaknesses in the implementation of critical cryptographic functions, preventing, for example, the use of weak cryptographic algorithms and key sizes. They also enable lower processing times and lower power consumption when performing cryptographic operations, offloading those computations from Cortex-M33. This is especially true for asymmetric cryptography.

For product certification purposes, ST can provide certified device information on how these security functions are implemented and validated.

For more information on crypto engine processing times, refer to their respective sections in the reference manual.

4.8.1 Crypto engines features

Table 19 lists the accelerated cryptographic operations available in the devices. One AES accelerators is available (it can be reserved to secure application only).

Note: Additional operations can be added using firmware.

The PKA can accelerate asymmetric crypto operations (like key pair generation, ECC scalar multiplication, point on curve check). See *Section 27: Public key accelerator (PKA)* for more details.

Table 19. Accelerated cryptographic operations

Operations	Algorithm	Specification	Key length (in bit)	Modes
Get entropy	RNG	NIST SP800-90B(1)	N/A	Software and hardware(2) modes running in parallel
Encryption, decryption	AES	FIPS PUB 197 NIST SP800-38A	128, 256	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D		GCM, CCM
Cipher-based message authentication code		NIST SP800-38D		GMAC
Checksum	SHA-1	FIPS PUB 180-4	N/A	Digest 160-bit
Cryptographic hash	SHA2-224	FIPS PUB 180-4		Digest 224-bit
	SHA2-256			Digest 256-bit
Keyed-hashing for message authentication	HMAC	FIPS PUB 198-1 IETF RFC 2104	Short, long (>64 bytes)	-
Encryption/decryption key-pair generation	RSA	IETF RFC 8017 NIST SP800-56B	Up to 4160	RSAES-OAEP
Signature with hashing Signature verification	RSA	IETF RFC 8017 FIPS PUB 186-4	Up to 4160	PKCS1-v1_5, PSS
	ECDSA	ANSI X9.62 IETF RFC 7027 FIPS PUB 186-4	Up to 640	*Table 235: Family of supported curves for ECC operations*
Key agreement	ECDH	ANSI X9.42		

1. Certifiable using STMicroelectronics reviewed documents.
2. Random numbers distribution to PKA using a dedicated hardware bus.

Note: Binary curves and Edwards curves are not supported by the PKA.

4.8.2 On-the-fly decryption engine (OTFDEC)

The OTFDEC TrustZone-aware peripheral features on-the-fly decryption of encrypted images stored in external memory, connected through XSPI.

The OTFDEC can also be used to encrypt clear text images stored in external memory. This can be used to encrypt an image with a device unique secret key.

On a tamper event all OTFDEC keys are erased and encrypted regions are read as zero until proper OTFDEC reinitialization.

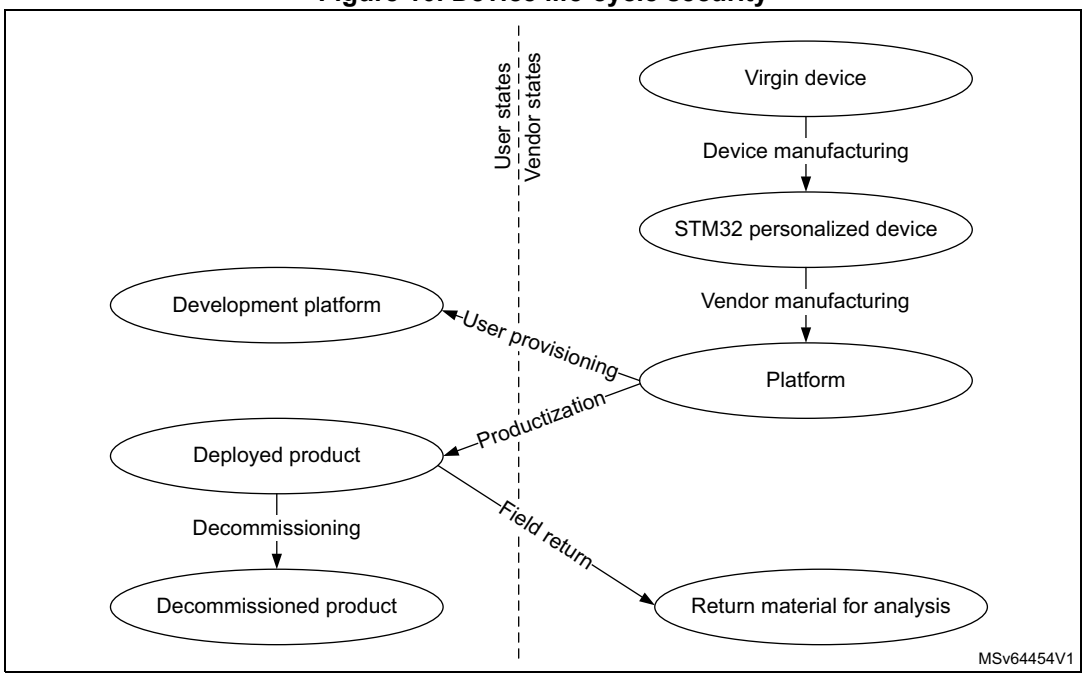
An OTFDEC typical use is detailed in Section 4.11.2: Software intellectual property protection with OTFDEC.

For more details on the OTFDEC refer to Section 26: On-the-fly decryption engine (OTFDEC).

4.9 Product life cycle

A typical IoT device life cycle is summarized in *Figure 10*. For each step, the devices propose secure life-cycle management mechanisms embedded in the hardware.

Figure 10. Device life-cycle security



More details on the various phases and associated transitions, found either at the vendor or end-user premises, are summarized in *Table 20*.

Table 20. Main product life-cycle transitions

Transitions	Description
Device manufacturing	STMicroelectronics creates new STM32 devices, always checking for manufacturing defects. During this process STM32 is provisioned with ROM firmware, secure firmware , and a public ID.
Vendor manufacturing	One (or more) vendor is responsible for the platform assembly, initialization, and provisioning before delivery to the end user. This end user can use the final product (“productization” transition) or use the platform for software development (“user provisioning” transition).
Productization	The end user gets a product ready for use. All security functions of the platform are enabled, the debugging/testing features are restricted/disabled, and unique boot entry to immutable code is enforced.
User provisioning	Platform vendor prepares an individual platform for development, not to be connected to a production cloud network.
Field return or decommissioning	Those are one-way transitions, with devices kept in user premises or returned to the manufacturer. In both cases, all data including user data are destroyed, therefore the devices loose the ability to operate securely (like connecting to a managed IoT network).

The features described hereafter contribute to secure the device life cycle.

4.9.1 Life cycle management with readout protection (RDP)

The readout protection mechanism (full hardware feature) controls the access to the devices debug, test and provisioned secrets, as summarized in *Table 21*.

Table 21. Typical product life-cycle phases

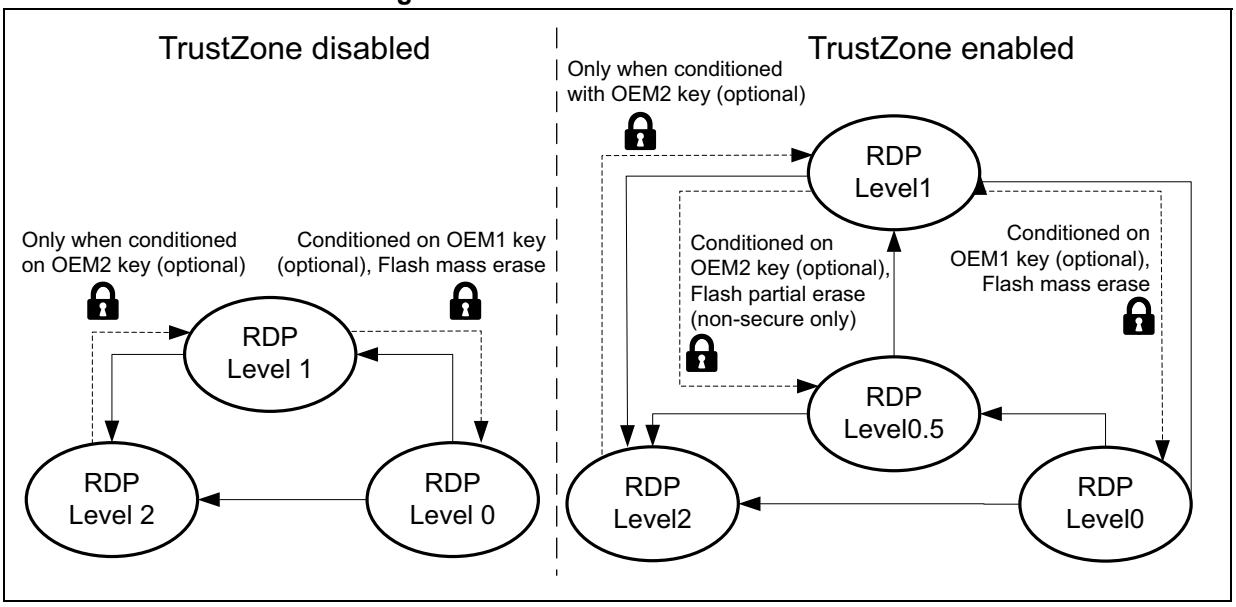
RDP protection level		Debug	Comments
0	Device open	Secure(1) and nonsecure	Boot address must target a secure area when TrustZone is enabled (secure SRAM, secure flash memory, RSS in system flash memory). Both OEM1 and OEM2 unlocking keys can be provisioned in the User options.
0.5(2)	Device partially closed (closed secure)	Nonsecure only	Boot address must target a secure area when TrustZone is enabled (secure user or system flash memory). Boot on SRAM is not permitted. Access to nonsecure flash memory is allowed when debug is connected. Both OEM1 and OEM2 unlocking keys can be provisioned in the User options.
1	Device memories protected	Nonsecure only (conditioned)	Boot address must target the secure user flash memory. Accesses to nonsecure flash memory, encrypted flash memory(3), SRAM2 and backup registers are not allowed when debug is connected. Both OEM1 and OEM2 unlocking keys can be provisioned in the User options.
2	Device closed	None (JTAG fuse)	Boot address must target the user flash memory (secure if TZEN = 1). Option bytes are read-only, hence RDP level 2 cannot be changed, unless OEM2 unlocking key is activated (see *Table 22*).

1. Debug is not available when executing RSS code.
2. Only applicable when TrustZone security is activated in the product.
3. External flash memory area decryption on-the-fly with OTFDEC.

Note: OEM1KEY option byte can be modified when OEM1LOCK = 0 (RDP = 0.5 or 1 only).
OEM2KEY option byte can be modified when OEM2LOCK = 0 (RDP = 1 only).

The supported transitions, summarized in *Figure 11*, can be requested (when available) through the debug interface or via the system bootloader.

Figure 11. RDP level transition scheme



The user flash memory is automatically erased, either partially or in totality, during an RDP regression from level 1. Those regressions can be conditioned to dedicated 64-bit password keys, if provisioned by the OEM (see next subsection). During the regression from RDP Level1 to RDP Level 0.5, only nonsecure embedded flash memory is erased, keeping functional, for example, the secure boot and the secure firmware update. In all regressions from RDP Level 1 the OTP area in the flash memory is kept, and targeted device secrets are erased. No secrets must be stored in the OTP as they are nonsecure. The same secrets are erased, as those erased as response to tamper, defined in *Section 4.6.3*.

Note: Enabling TrustZone using the option byte TZEN is possible only when the RDP level is 0.

For more details on RDP, refer to *Section 7: Embedded flash memory (FLASH)*.

RDP regression and erase

When regressing RDP to level 0.5 or level 0, the SRAM1, SRAM2, PKA SRAM memories, and TAMP backup registers are erased.

Note: Before launching an RDP regression, the software must invalidate the ICACHE and wait for the BUSYF bit to get cleared.

RDP unlocking sequences

The use of the two OEM password keys described in *Figure 11* is further described hereafter.

Note: The devices support both permanent RDP Level 2 (legacy mode) or password-based RDP Level 2 regression to Level 1. This Level 2 regression does not erase the application code, and it does not change the RDP Level 1 protections in place.

Details on the password-based regression can be found in *Table 22*.

Table 22. OEM key RDP unlocking methods

OEM1 password options			OEM2 password options		
OEM1 LOCK	**Initial RDP level**	**RDP regression**	**OEM2 LOCK**	**Initial RDP level**	**RDP regression**
1	1	Regression to Level 0 possible only through OEM1 unlock sequence (see below)	1	1	Regression to Level 0.5 possible only through OEM2 unlock sequence A (see below)
				2	Automatic regression to Level 1 triggered upon successful OEM2 unlock sequence B (see below)
0		Regression to Level 0 always granted	0	1	Regression to Level 0.5 always granted
				2	Regression to Level 1 never granted. RDP level 2 remains a permanent state.

- OEM1 unlock sequence, starting at RDP Level 1:
  - Shift password key through JTAG/SWD (see the note below).
  - If this key matches the OEM1KEY provisioned in the device, the application can trigger a regression sequence to Level 0. After the regression is completed, the nonsecure embedded flash memory and device secrets are erased. The OTP area is not erased.
  - In case of mismatch value, the RDP regression is blocked and RDP Level 1 protections are enforced until the next power-on reset.
- OEM2 unlock sequence A, starting at RDP Level 1:
  - Shift password key through JTAG/SWD under reset (see the note below).
  - If this key matches the OEM2KEY provisioned in the device, the application can trigger a regression sequence to Level 0.5. After the regression is completed, the nonsecure part of the embedded flash memory and the device secrets are erased. The OTP area is not erased.
  - In case of mismatch value, the RDP regression is blocked and RDP Level 1 protections are enforced until the next power-on reset.
- OEM2 unlock sequence B, starting at RDP Level 2:
  - Shift password key through JTAG/SWD under reset (see the note below).
  - If this key matches the OEM2KEY provisioned in the device, the device automatically triggers a regression sequence to Level 1. After the regression is completed, a power-on reset has to be applied.
  - In case of mismatch value, the RDP regression is blocked and RDP Level 2 protections are enforced until the next power-on reset.

Note: Unlocking the device with a password is possible only once per power cycle.

Shifting the password key through JTAG/SWD corresponds to writing four 32-bit key words, AUTH_KEY[31:0], followed by AUTH_KEY[63:32], AUTH_KEY[95:64], and last AUTH_KEY[127:96] in DBGMCU_DBG_AUTH_HOST register.

JTAG 32-bit device specific ID

Unless the JTAG port is deactivated (OEM2LOCK = 0 and RDP Level = 2), a 32-bit device specific quantity can always be read through the JTAG port. This information is stored in DBGMCU_DBG_AUTH_DEVICE.

The OEM can use this 32-bit information to derive the expected OEM password keys to unlock this specific device.

4.9.2 Recommended option byte settings

Most of the time, the user threat model focuses mainly on software attacks. In this case, it may be sufficient to keep the RDP Level 1 as device protection.

For a more aggressive threat model, where the user fears physical attacks on the STM32 device, it is recommended to optimize the level of security by setting the RDP Level 2.

The recommended settings are detailed below:

- If TrustZone is disabled (TZEN = 0)
  - RDP Level 2
  - nonsecure boot address option bytes set in user flash memory
- If TrustZone is enabled (TZEN = 1)
  - RDP Level 2
  - BOOT_LOCK = 1
  - secure boot address option bytes set in user secure flash memory

As described in the previous section, the customer can decide to allow any RDP Level 2 part to regress to RDP Level 1, provided the OEM2 key has been successfully provisioned, and the OEM2LOCK option bit is set.

4.10 Access controlled debug

The device restricts access to embedded debug features, to guarantee the confidentiality of customer assets against unauthorized usage of debug and trace features.

4.10.1 Debug protection with readout protection (RDP)

As described in *Section 4.9.1,* the hardware RDP mechanism automatically controls the accesses to the device debug and test. The protection of these debug features is defined in *Table 23*. Possible password-based regressions are described in *Section 4.9.1*.

Table 23. Debug protection with RDP

RDP protection level		Debug features protection
0	Device open	Any debug(1)
0.5(2)	Device partially closed	Secure debug is no longer available.
1	Device memories protected	Nonsecure debug can no longer debug code and data stored in the embedded flash memory, encrypted external flash memory(3),SRAM2, and backup registers.
2	Device closed	JTAG is physically deactivated, unless it is kept operational only for password key injection (OEM2LOCK = 1). See *Section 4.9.1*.

1. Including ST engineering test modes, used for field returns.
2. Only applicable when TrustZone security is activated in the product.
3. External flash memory area decryption on-the-fly with OTFDEC.

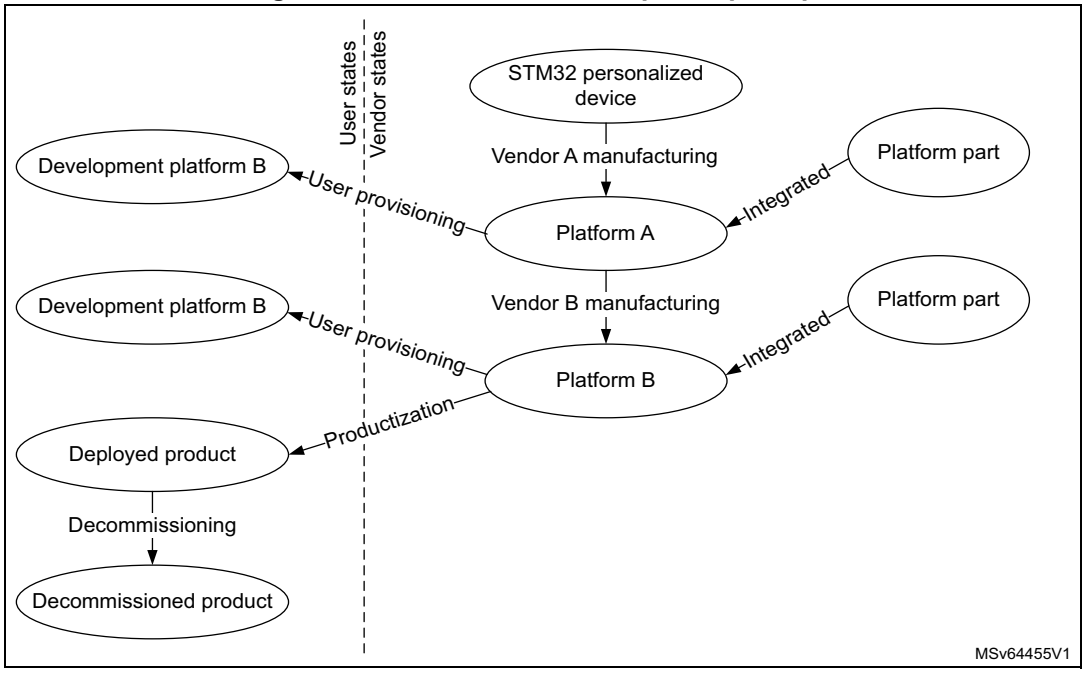
4.11 Software intellectual property protection and collaborative development

Thanks to software intellectual property protection and collaborative model, the devices allow the design of solutions integrating innovative third-party libraries.

Collaborative development is summarized *Figure 12*. Starting from a personalized device sold by STMicroelectronics, a vendor A can integrate a portion of hardware and software on a platform A, which can then be used by a vendor B, who does the same before deploying a final product to the end users.

Note: Each platform vendor can provision individual platforms for development not to be connected to a production cloud network ("Development Platform X").

Figure 12. Collaborative development principle



The features described hereafter contribute to securing the software intellectual property within such a collaborative development.

4.11.1 Software intellectual property protection with RDP

As described in *Section 4.9.1,* the hardware RDP mechanism automatically controls the accesses to secrets provisioned in the device. The protection of these secrets is defined in *Table 24*.

Table 24. Software intellectual property protection with RDP

RDP protection level		Secrets protection
Level 0	Device open	No special protections.
Level 0.5(1)	Device partially closed	All peripherals and memories mapped as secure during secure boot cannot be dumped, debugged or traced
Level 1	Device memories protected	Data and code stored in embedded flash memory, encrypted external memory(2), SRAM2, and backup registers are no more accessible via debugger.
Level 2	Device closed	All data and code stored in the device or encrypted in external memory cannot be dumped, debugged or traced.

1. Only applicable when TrustZone® security is activated in the product.
2. External flash memory area decryption on-the-fly with OTFDEC.

4.11.2 Software intellectual property protection with OTFDEC

As described in *Section 26: On-the-fly decryption engine (OTFDEC),* the OTFDEC associated with the XSPI is able to decrypt on-the-fly, the encrypted images stored in external memories.

Thanks to this feature, the devices allow the installation of intellectual properties in one of the following ways:

- Image already encrypted with a key provisioned in the device
- Clear text images, encrypted in place

Note: *Since the OTFDEC uses the AES in counter mode (CTR) to achieve the lowest possible latency, each time the content of one encrypted region is changed, the corresponding cryptographic context (key or initialization vector) must be changed. This constraint makes OTFDEC suitable to decrypt read-only data or code stored in external memory.*

Secure boot

After provisioning, each time the device initializes on a trusted firmware, the following actions are required:

1. Secure boot firmware executes, programming the external firmware and data AES keys to the OTFDEC write-only key registers, along with the other needed information.
2. The application reads or executes the encrypted firmware and data in external memory. In this case of tamper event detection, all OTFDEC keys are erased, and encrypted regions are read as zero until the OTFDEC is properly reinitialized again.

4.11.3 Other software intellectual property protections

The device additional protections to software intellectual property are:

- Invasive attacks such as physical tampering are countered by detection then decommissioning of the device before the detected attack succeeds.

Noninvasive attacks, such as side channel attacks, are countered by not leaking secret information via side channels such as timing, power, and EM emissions.

5 Global TrustZone® controller (GTZC)

5.1 Introduction

This section describes the global TrustZone controller (GTZC) block, containing the following subblocks:

- **TZSC**: TrustZone security controller

  This subblock defines the secure/privileged state of slave peripherals. It also controls the subregions size and properties for the watermark memory protection controller (MPCWM). The TZSC informs some peripherals (such as RCC or GPIOs) about the secure status of each securable peripheral.
- **MPCBB**: memory protection controller - block based

  This subblock configures the internal RAM in a TrustZone-system product having segmented SRAM (pages of 512 bytes) with programmable-security and privileged attributes.
- **TZIC**: TrustZone illegal access controller

  This subblock gathers all illegal access events in the system and generates a secure interrupt towards NVIC.

These subblocks are used to configure TrustZone system security in a product having bus agents with programmable-security and privileged attributes such as:

- On-chip RAM with programmable secure and/or privileged blocks (pages)
- AHB and APB peripherals with programmable security and/or privileged access
- Off-chip memories with secure and/or privileged areas

5.2 GTZC main features

The GTZC main features are listed below:

- Independent 32-bit AHB interface for TZSC, TZIC, and MPCBB
- TZIC accessible only with secure transactions
- Secure and nonsecure access supported for privileged and unprivileged part of TZSC and MPCBB
- Set of registers to define product security settings:
  - Secure/privileged blocks for internal SRAM
  - Secure/privileged regions for external memories
  - Secure/privileged access mode for securable peripherals

GTZC TrustZone system architecture

The Armv8-M supports security per TrustZone-M model with isolation between:

- A secure world, where usually security sensitive applications are run and critical resources are located
- A nonsecure or public world (such as usual nonsecure operating system and user space)

The TrustZone architecture is extended beyond AHB and Armv8-M with:

- AHB/APB bridge used as a secure gate to block or propagate secure/nonsecure and privileged/unprivileged transaction towards APB agents
- PPC (peripheral protection controller) used as a secure gate to block or propagate secure/nonsecure and privileged/unprivileged transaction towards AHB agents
- TrustZone block-based MPC firewalls used as secure gates to filter secure/nonsecure, privileged/unprivileged access towards internal SRAMs
- TrustZone watermark MPC firewalls used as secure gates to filter secure/nonsecure, privileged/unprivileged access towards external memories

AHB and APB peripherals can be categorized as:

- **Privileged**: peripherals protected by an AHB/APB firewall stub that is controlled from TZSC to define privilege properties
- **Secure**: peripherals are always protected by an AHB/APB firewall stub. These peripherals are always secure (such as TZIC)
- **Securable**: peripherals protected by an AHB/APB firewall stub that is controlled from TZSC to define security properties (optional)
- **Nonsecure and unprivileged:** peripherals connected directly to AHB/APB interconnect without any secure gate
- **TrustZone-aware**: peripherals connected directly to AHB or APB bus and implementing a specific TrustZone behavior (such as a subset of registers being secure). TrustZone-aware AHB masters always drive HNONSEC signal according to their security mode (such has Armv8-M core or DMA)

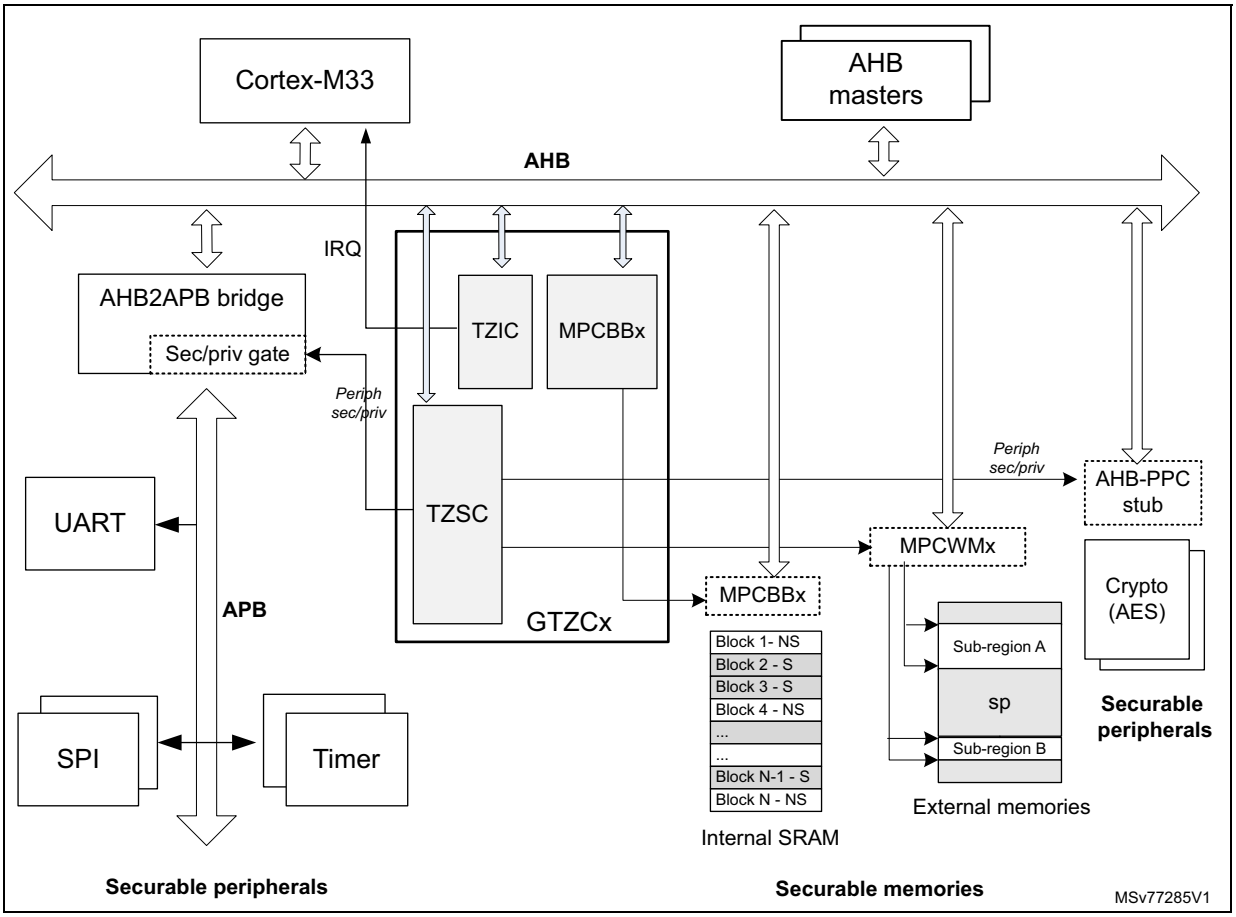
Application information

The TZSC (MPCWM), TZIC, and MPCBB can be used in one of the following ways:

- Programmed during secure boot only, locked and not changed afterwards
- Dynamically reprogrammed when using specific application code or secure kernel (microvisor). When not locked, MPCWM regions and MPCBB secure blocks can be changed by secure software executing from the secure FLASH region or secure SRAM. The same remark applies to the GTZC1_TZSC_SECCFGRn/PRIVCFGRn registers that define the secure/privileged state of each peripheral.

The Armv8-M security architecture with secure, securable, and TrustZone-aware peripherals is shown in the figure below.

Figure 13. GTZC in Armv8-M subsystem block diagram



5.3 GTZC implementation

The devices embed the following GTZC features.

Table 25. GTZC features

GTZC sub-blocks	GTZC
TZSC	X
TZIC	X
MPCWM (number of MPCWM)	X (1)
MPCBB (number of MPCBB)	X (2)

The tables below show the GTZC subblocks address offset versus GTZC base address (refer to *Section 2.3.2: Memory map and register boundary addresses* for GTZC base address).

Table 26. GTZC sub-blocks address offset

GTZC sub-blocks	Address offset
GTZC1_TZSC (including MPCWM)	0x0000
GTZC1_TZIC	0x0400
GTZC1_MPCBB1	0x0800
GTZC1_MPCBB2	0x0C00

The table below describes the characteristics of the available MPCWM.

Table 27. MPCWM resource assignment

MPC	Resource	Number of regions	Watermark granularity (bytes)	STM32WBA23	STM32WBA25
MPCWM1	XSPI1 memory	2	128 K	-	X

The table below describes the characteristics of the available MPCBB.

Table 28. MPCBB resource assignment

MPC	Resource	Memory size (Kbytes)	Block size (bytes)	Number of blocks	Number of super-blocks
MPCBB1	SRAM1	64	512	128	4
MPCBB2	SRAM2	32	512	64	2

5.4 GTZC functional description

5.4.1 GTZC block diagram

Figure 14 describes the combined feature of TZSC (MPCWM), TZIC, and MPCBB. Each subblock is controlled by its own AHB configuration port.

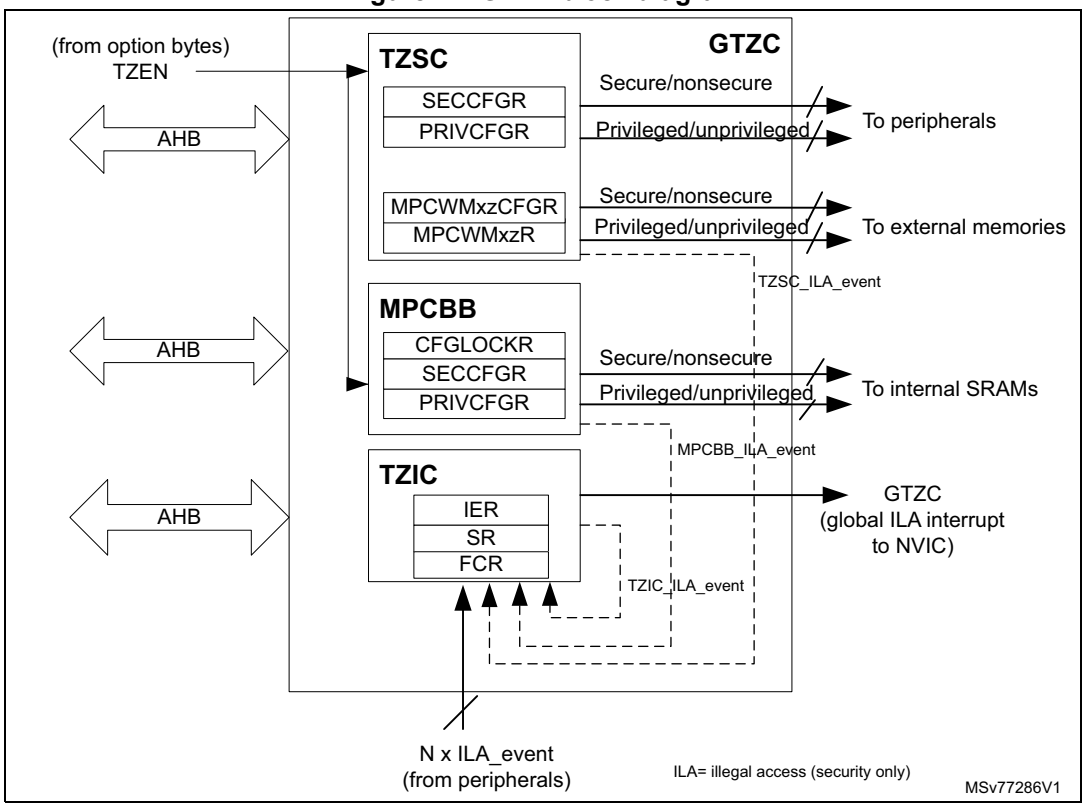
The TZSC defines which peripheral is secured and/or privileged. The privileged configuration bit of a peripheral can be modified by a secure privileged transaction when the peripheral is configured as secure. Otherwise, a privileged transaction (nonsecure) is sufficient.

On the opposite, the secure configuration bit of a peripheral can be modified only with a secure privileged transaction if the peripheral is configured as privileged. Otherwise, a secure transaction (unprivileged) is sufficient.

The MPCBB secure configuration bit of a given SRAN block can be modified only with a secure privileged transaction if the same SRAM block is configured as privileged. Otherwise, a secure transaction (unprivileged) is sufficient.

The TZIC gathers illegal events generated within the system when an illegal access is detected. TZIC can then generate a secure interrupt towards the CPU if needed.

Figure 14. GTZC block diagram



5.4.2 Illegal access definition

Three different types of illegal access (ILA) exist:

- Illegal nonsecure access

  Any nonsecure transaction trying to write a secure resource is considered as illegal: consequently, the addressed resource generates an illegal access interrupt for illegal write access and a bus error for illegal fetch access. Some exceptions exist on secure and privileged configuration registers: the latter ones authorize nonsecure read access

to secure registers (see GTZC1_TZSC_SECCFGRn and GTZC1_TZSC_PRIVCFGRn).

- Illegal secure access

  Any secure transaction trying to access a nonsecure block in internal block-based SRAM or watermarked memory is considered as illegal.

  Correct TZIC settings enable the capture of the associated event and then generate the GTZC1_TZIC interrupt to the NVIC. This applies to read, write, and execute accesses.

  Concerning the MPCBB controller, there is an option to ignore secure data read/write access on nonsecure SRAM blocks, by setting the SRWILADIS bit in the GTZC1_MPCBBx_CR register. Secure read and write data transactions are then allowed on nonsecure SRAM blocks, while secure execution access remains not allowed.

  Any secure execute transaction trying to access a nonsecure peripheral register is considered as illegal and generates a bus error.

- Illegal unprivileged access

  Any unprivileged transaction trying to access a privileged resource is considered as illegal. There is no illegal access event generated for illegal read and write access. The addressed resource follows a silent-fail behavior, returning all zero data for read and ignoring any write. No bus error is generated. A bus error is generated when any unprivileged execute transaction tries to access a privileged memory.

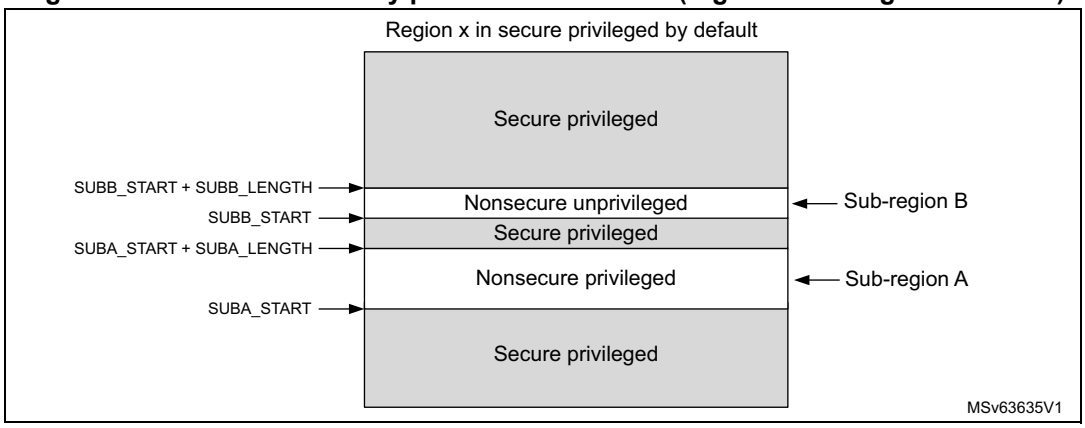
5.4.3 TrustZone security controller (TZSC)

The TZSC is composed of a configurable set of registers, providing the following features:

- The control of secure and privileged state for all peripherals is done through:
  - GTZC1_TZSC_SECCFGRn registers to control AHB/APB firewall stubs for the securable peripherals
  - GTZC1_TZSC_PRIVCFGRn registers to control AHB/APB firewall stubs for the privileged peripherals
- For watermark memory protection controller (external memories and backup SRAM), two independent regions can be defined and the following fields are used to program:
  - The start of the first protected subregion on external memory/backup SRAM: SUBA_START[10:0]
  - The length of the first protected subregion on external memory/backup SRAM: SUBA_LENGTH[11:0]
  - The start of the second protected subregion on external memory/backup SRAM: SUBB_START[10:0]
  - The length of the second protected subregion on external memory/backup SRAM: SUBB_LENGTH[11:0]

A control register for each subregion can be used to enable/disable the watermark memory protection controller as well as defining the right attributes of each subregion.

Figure 15. Watermark memory protection controller (region x/sub-regions A and B)



In the figure above, region x represents the external memory region (such as XSPI1 bank). Secure and privileged attributes of subregions A and B are independently configurable. When no subregions are defined or enabled on the region x, then the default attribute of the region x is set as “secure-privileged”.

The tables below describe the secure/privileged properties of the common area of subregion A and B when an overlap exists.

Table 29. Secure properties of sub-regions A and B

Sub-region A	Sub-region B	Properties of overlapped region A and B
Nonsecure	Nonsecure	Nonsecure
Nonsecure	Secure	Nonsecure
Secure	Nonsecure	Nonsecure
Secure	Secure	Secure

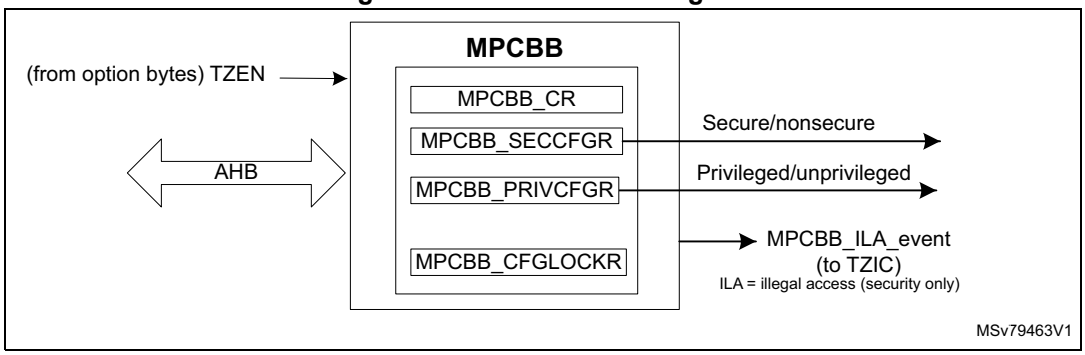
Table 30. Privileged properties of sub-regions A and B

Sub-region A	Sub-region B	Properties of overlapped region A and B
Unprivileged	Unprivileged	Unprivileged
Unprivileged	Privileged	Unprivileged
Privileged	Unprivileged	Unprivileged
Privileged	Privileged	Privileged

5.4.4 Memory protection controller - block based (MPCBB)

The MPCBB is composed of a configurable set of registers allowing security and privileged policy to be defined for internal SRAM memories. The security and privileged policy can be individually configured per each 512-byte block of SRAM.

Figure 16. MPCBB block diagram



In order to set up the MPCBB, the following actions are needed (for example at boot time):

- Secure firmware must define which memory blocks are secure by setting the correct bits in GTZC1_MPCBBx_SECCFGRn.
- Privileged firmware must define which memory blocks are privileged by setting the correct bits in GTZC1_MPCBBx_PRIVCFGRn.

An MPCBB superblock is made of 32 consecutive blocks. For each superblock, a secure application can lock all related secure/privileged bits using the correct bits in GTZC1_MPCBBx_CFGLOCK. This lock remains active until the next system reset.

Note: The block size is 512 bytes. The superblock size is 512 x 32 = 16 Kbytes.

5.4.5 TrustZone illegal access controller (TZIC)

The TZIC concentrates all illegal access source events. It is used only when the system is TrustZone enabled (TZEN = 1).

TZIC allows the trace (flag) of which event triggered the secure illegal access interrupt. Register masks (GTZC1_TZIC_IERx) are available to filter unwanted events. On an unmasked illegal event, TZIC generates the GTZC1_IRQn interrupt to the NVIC.

For each illegal event source, a status flag and a clear bit exist (respectively within GTZC1_TZIC_SRx and GTZC1_TZIC_FCRx). The reset value of mask registers (GTZC1_TZIC_IERx) is such that all events are masked.

5.4.6 Power-on/reset state

The power-on and reset state of the TZSC clear to 0 all bits of GTZC1_TZSC_SECCFGRn and GTZC1_TZSC_PRIVCFGRn, meaning that all securable peripherals are respectively set to nonsecure and unprivileged.

For internal SRAMs, all GTZC1_MPCBBx_SECCFGRn and GTZC1_MPCBBx_PRIVCFGRn are set:

- When TrustZone security is enabled at system level (TZEN = 1), to 0xFFFF FFFF, making these internal memories block secure and privileged by default
- When TrustZone security is disabled at system level (TZEN = 0), to 0x0000 0000, making these internal memories block nonsecure and unprivileged by default

For external memories MPCWMx:

- All GTZC1_TZSC_MPCWMxyR registers are set to 0x0000 0000, making these memories secure and privileged by default when TrustZone security is enabled at system level (TZEN = 1)
- When TrustZone security is disabled at system level (TZEN = 0), GTZC1_TZSC_PMCWMxyR registers are not accessible and these memories are nonsecure and unprivileged.

Secure boot code can then program the security settings, making components secure or not as needed.

5.5 GTZC interrupts

TZIC is a secure peripheral, which systematically generates an illegal access event when accessed by a nonsecure access. The MPCBB and TZSC are TrustZone-aware peripherals, meaning that secure and nonsecure registers coexist within the peripheral.

Table 31. GTZC interrupt request

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop mode	Exit Standby mode
GTZC	Illegal access	All flags in GTZC1_TZIC_SRx	All bits in GTZC1_TZIC_IERx	Write 1 in the bit GTZC1_TZIC_FCRx	Yes	No	No

5.6 GTZC1 TZSC registers

All registers can only be accessed in words (32-bit).

5.6.1 GTZC1 TZSC control register (GTZC1_TZSC_CR)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged access only.

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

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **LCK**: Lock the configuration of GTZC1_TZSC_SECCFGRn and GTZC1_TZSC_PRIVCFGRn registers until next reset

This bit is cleared by default and once set, it can not be reset until system reset.

0: Configuration of all GTZC1_TZSC_SECCFGRn and GTZC1_TZSC_PRIVCFGRn registers not locked

1: Configuration of all GTZC1_TZSC_SECCFGRn and GTZC1_TZSC_PRIVCFGRn registers locked

5.6.2 GTZC1 TZSC secure configuration register 1 (GTZC1_TZSC_SECCFGR1)

Address offset: 0x010

Reset value: 0x0000 0000

Write-secure access only.

This register can be written only by secure privileged transaction when the corresponding GTZC1_TZSC_PRIVCFGR1 register bit is set to 1. If a given PRIV bit is not set, the equivalent SEC bit can be written by secure unprivileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2SEC	Res.
														rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	I2C1SEC	Res.	Res.	Res.	Res.	Res.	IWDGSEC	Res.	Res.	Res.	Res.	Res.	Res.	TIM2SEC
		rw						rw							rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **LPTIM2SEC**: Secure access mode for LPTIM2

0: Nonsecure

1: Secure

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **I2C1SEC**: Secure access mode for I2C1

0: Nonsecure

1: Secure

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **IWDGSEC**: Secure access mode for IWDG

0: Nonsecure

1: Secure

Bits 6:1 Reserved, must be kept at reset value.

Bit 0 **TIM2SEC**: Secure access mode for TIM2

0: Nonsecure

1: Secure

5.6.3 GTZC1 TZSC secure configuration register 2 (GTZC1_TZSC_SECCFGR2)

Address offset: 0x014

Reset value: 0x0000 0000

Write-secure access only.

This register can be written only by a secure privileged transaction when the corresponding GTZC1_TZSC_PRIVCFGR2 register bit is set to 1. If a given PRIV is not set, the equivalent SEC bit can be written by a secure unprivileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4SEC	Res.	Res.	Res.	Res.	LPTIM1SEC	I2C3SEC	LPUART1SEC	SPI3SEC
							rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USB1SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1SEC	TIM17SEC	TIM16SEC	Res.	USART1SEC	Res.	Res.	Res.
rw								rw	rw	rw		rw			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **ADC4SEC**: Secure access mode for ADC4
0: Nonsecure
1: Secure

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **LPTIM1SEC**: Secure access mode for LPTIM1
0: Nonsecure
1: Secure

Bit 18 **I2C3SEC**: Secure access mode for I2C3
0: Nonsecure
1: Secure

Bit 17 **LPUART1SEC:** Secure access mode for LPUART1
0: Nonsecure
1: Secure

Bit 16 **SPI3SEC:** Secure access mode for SPI3
0: Nonsecure
1: Secure

Bit 15 **USB1SEC:** Secure access mode for USB1
0: Nonsecure
1: Secure
Note: This bit is reserved on STM32WBA23xx devices

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **SAI1SEC:** Secure access mode for SAI1
0: Nonsecure
1: Secure

Bit 6 **TIM17SEC**: Secure access mode for TIM17
0: Nonsecure
1: Secure

Bit 5 **TIM16SEC**: Secure access mode for TIM16
0: Nonsecure
1: Secure

Bit 4 Reserved, must be kept at reset value.

Bit 3 **USART1SEC**: Secure access mode for USART1
0: Nonsecure
1: Secure

Bits 2:0 Reserved, must be kept at reset value.

5.6.4 GTZC1 TZSC secure configuration register 3 (GTZC1_TZSC_SECCFGR3)

Address offset: 0x018

Reset value: 0x0000 0000

Write-secure access only.

This register can be written only by a secure privileged transaction when the corresponding GTZC1_TZSC_PRIVCFGR3 register bit is set to 1. If a given PRIV is not set, the equivalent SEC bit can be written by a secure unprivileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTACONVSEC	RADIOSEC	RAMCFGSEC	Res.	XSPI1SEC	Res.	Res.	Res.	PKASEC
							rw	rw	rw		rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	RNGSEC	HASHSEC	AESSEC	Res.	Res.	Res.	Res.	ICACHE_REGSEC	Res.	Res.	CRCSEC	Res.	Res.	Res.
		rw	rw	rw					rw			rw			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **PTACONVSEC**: Secure access mode for PTACONV
0: Nonsecure
1: Secure

Bit 23 **RADIOSEC**: Secure access mode for 2.4 GHz RADIO
0: Nonsecure
1: Secure

Bit 22 **RAMCFGSEC**: Secure access mode for RAMCFG
0: Nonsecure
1: Secure

Bit 21 Reserved, must be kept at reset value.

Bit 20 **XSPI1SEC**: Secure access mode for XSPI1 and DLYBXS1
0: Nonsecure
1: Secure

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **PKASEC**: Secure access mode for PKA
0: Nonsecure
1: Secure

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **RNGSEC**: Secure access mode for RNG
0: Nonsecure
1: Secure

Bit 12 **HASHSEC**: Secure access mode for HASH
0: Nonsecure
1: Secure

Bit 11 **AESSEC**: Secure access mode for AES
0: Nonsecure
1: Secure

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 **ICACHE_REGSEC**: Secure access mode for ICACHE registers
0: Nonsecure
1: Secure

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **CRCSEC**: Secure access mode for CRC
0: Nonsecure
1: Secure

Bits 2:0 Reserved, must be kept at reset value.

5.6.5 GTZC1 TZSC privilege configuration register 1 (GTZC1_TZSC_PRIVCFGR1)

Address offset: 0x020

Reset value: 0x0000 0000

Write-privileged access only.

This register can be written only by a secure privileged transaction when the corresponding GTZC1_TZSC_SECCFGR1 register bit is set to 1. If a given SEC bit is not set, the equivalent PRIV bit can be written by a nonsecure privileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2PRIV	Res.
														rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	I2C1PRIV	Res.	Res.	Res.	Res.	Res.	IWDGPRIV	Res.	Res.	Res.	Res.	Res.	Res.	TIM2PRIV
		rw						rw							rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **LPTIM2PRIV**: Privileged access mode for LPTIM2
- 0: Unprivileged
- 1: Privileged

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **I2C1PRIV**: Privileged access mode for I2C1
- 0: Unprivileged
- 1: Privileged

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **IWDGPRIV**: Privileged access mode for IWDG
- 0: Unprivileged
- 1: Privileged

Bits 6:1 Reserved, must be kept at reset value.

Bit 0 **TIM2PRIV**: Privileged access mode for TIM2
- 0: Unprivileged
- 1: Privileged

5.6.6 GTZC1 TZSC privilege configuration register 2 (GTZC1_TZSC_PRIVCFGR2)

Address offset: 0x024

Reset value: 0x0000 0000

Write-privileged access only.

This register can be written only by a secure privileged transaction when the corresponding GTZC1_TZSC_SECCFGR2 register bit is set to 1. If a given SEC bit is not set, the equivalent PRIV bit can be written by a nonsecure privileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4PRIV	Res.	Res.	Res.	Res.	LPTIM1PRIV	I2C3PRIV	LPUART1PRIV	SPI3PRIV
							rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1PRIV	TIM17PRIV	TIM16PRIV	Res.	USART1PRIV	Res.	Res.	Res.
rw								rw	rw	rw		rw			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **ADC4PRIV**: Privileged access mode for ADC4
0: Unprivileged
1: Privileged

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **LPTIM1PRIV**: Privileged access mode for LPTIM1
0: Unprivileged
1: Privileged

Bit 18 **I2C3PRIV**: Privileged access mode for I2C3
0: Unprivileged
1: Privileged

Bit 17 **LPUART1PRIV**: Privileged access mode for LPUART1
0: Unprivileged
1: Privileged

Bit 16 **SPI3PRIV**: Privileged access mode for SPI3
0: Unprivileged
1: Privileged

Bit 15 **USBPRIV**: Privileged access mode for USB
0: Unprivileged
1: Privileged
Note: This bit is reserved on STM32WBA23xx devices

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **SAI1PRIV**: Privileged access mode for SAI1
0: Unprivileged
1: Privileged

Bit 6 **TIM17PRIV**: Privileged access mode for TIM17
0: Unprivileged
1: Privileged

Bit 5 **TIM16PRIV**: Privileged access mode for TIM16
0: Unprivileged
1: Privileged

Bit 4 Reserved, must be kept at reset value.

Bit 3 **USART1PRIV**: Privileged access mode for USART1
0: Unprivileged
1: Privileged

Bits 2:0 Reserved, must be kept at reset value.

5.6.7 GTZC1 TZSC privilege configuration register 3 (GTZC1_TZSC_PRIVCFGR3)

Address offset: 0x028

Reset value: 0x0000 0000

Write-privileged access only.

This register can be written only by a secure privileged transaction when the corresponding GTZC1_TZSC_SECCFGR3 register bit is set to 1. If a given SEC bit is not set, the equivalent PRIV bit can be written by a nonsecure privileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTACONVPRIV	RADIOPRIV	RAMCFGPRIV	Res.	XSPI1PRIV	Res.	Res.	Res.	PKAPRIV
							rw	rw	rw		rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	RNGPRIV	HASHPRIV	AESPRIV	Res.	Res.	Res.	Res.	ICACHE_REGPRIV	Res.	Res.	CRCPRIV	Res.	Res.	Res.
		rw	rw	rw					rw			rw			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **PTACONVPRIV**: Privileged access mode for PTACONV
0: Unprivileged
1: Privileged

Bit 23 **RADIOPRIV**: Privileged access mode for 2.4 GHz RADIO
0: Unprivileged
1: Privileged

Bit 22 **RAMCFGPRIV**: Privileged access mode for RAMCFG
0: Unprivileged
1: Privileged

Bit 21 Reserved, must be kept at reset value.

Bit 20 **XSPI1PRIV**: Privileged access mode for XSPI1 and DLYBXS1
0: Unprivileged
1: Privileged

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **PKAPRIV**: Privileged access mode for PKA
0: Unprivileged
1: Privileged

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **RNGPRIV**: Privileged access mode for RNG
0: Unprivileged
1: Privileged

Bit 12 **HASHPRIV**: Privileged access mode for HASH
0: Unprivileged
1: Privileged

Bit 11 **AESPRIV**: Privileged access mode for AES
0: Unprivileged
1: Privileged

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 **ICACHE_REGPRIV**: Privileged access mode for ICACHE registers
0: Unprivileged
1: Privileged

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **CRCPRIV**: Privileged access mode for CRC
0: Unprivileged
1: Privileged

Bits 2:0 Reserved, must be kept at reset value.

5.6.8 GTZC1 TZSC memory 1 subregion z watermark configuration register (GTZC1_TZSC_MPCWM1zCFGR) (z = A to B)

Address offset: Block A: 0x040
Address offset: Block B: 0x048
Reset value: 0x0000 0000

Secure privilege access only.

Note: Some registers are only available on some devices in the STM32WBA2. Refer to the device datasheet for availability of its associated memory region.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
						rw	rw							rs	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **PRIV**: Privileged sub-region z

This bit is taken into account only if SREN is set.

0: Privileged and unprivileged accesses are granted in sub-region z.

1: Only privileged accesses are granted in sub-region z.

Bit 8 **SEC**: Secure sub-region z

This bit is taken into account only if SREN is set.

0: Only nonsecure data accesses are granted to sub-region z.

1: Only secure data accesses are granted to sub-region z.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **SRLOCK**: Sub-region z lock

This bit, once set, can be cleared only by a system reset.

0: GTZC1_TZSC_MPCWM1zCFGR, GTZC1_TZSC_MPCWM1zR can be written.

1: Writes to GTZC1_TZSC_MPCWM1zCFGR, GTZC1_TZSC_MPCWM1zR are ignored.

Bit 0 **SREN**: Sub-region z enable

0: Sub-region z is disabled. Access control applies to any access between this sub-region start- and end-addresses.

1: Sub-region z is enabled. Access control defined in GTZC1_TZSC_MPCWM1zCFGR applies to any access between this sub-region start- and end-addresses, defined in GTZC1_TZSC_MPCWM1zR.

5.6.9 GTZC1 TZSC memory 1 subregion A watermark register (GTZC1_TZSC_MPCWM1AR)

Address offset: 0x044

Reset value: 0x0000 0000

Secure privilege access only.

When SUBA_START + SUBA_LENGTH is higher than the maximum size allowed for the memory, a saturation of SUBA_LENGTH is applied automatically.

When an overlap of subregion A and B exists, secure/privileged attributes of both sub-regions apply on the common section (see *Section 5.4.3*).

Note: Some registers are only available on some devices in the STM32WBA2 series. Refer to the device datasheet for availability of its associated memory region.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	SUBA_LENGTH[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SUBA_START[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **SUBA_LENGTH[11:0]**: Length of sub-region A

This field defines the length of the sub-region A, to be multiplied by the granularity defined in *Table 27*.

When SUBA_START + SUBA_LENGTH is higher than the maximum size allowed for the memory, a saturation of SUBA_LENGTH is applied automatically.

If SUBA_LENGTH = 0, the sub-region A is disabled. (SREN bit in GTZC1_TZSC_MPCMW1ACFGR is cleared).

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **SUBA_START[10:0]**: Start of sub-region A

This field defines the address offset of the sub-region A, to be multiplied by the granularity defined in *Table 27*, versus the start of memory bank base address.

5.6.10 GTZC1 TZSC memory 1 subregion B watermark register (GTZC1_TZSC_MPCWM1BR)

Address offset: 0x04C
Reset value: 0x0000 0000
Secure privilege access only.

When SUBB_START + SUBB_LENGTH is higher than the maximum size allowed for the memory, a saturation of SUBB_LENGTH is applied automatically.

When an overlap of subregion A and B exists, secure/privileged attributes of both subregions apply on the common section (see *Section 5.4.3*).

Note: Some registers are only available on some devices in the STM32WBA2. Refer to the device datasheet for availability of its associated memory region.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	SUBB_LENGTH[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SUBB_START[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **SUBB_LENGTH[11:0]**: Length of sub-region B
This field defines the length of the sub-region B, to be multiplied by the granularity defined in *Table 27*.
When SUBB_START + SUBB_LENGTH is higher than the maximum size allowed for the memory, a saturation of SUBB_LENGTH is applied automatically.
If SUBB_LENGTH = 0, the sub-region B is disabled. (SREN bit in GTZC1_TZSC_MPCMW1BCFGR is cleared).

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **SUBB_START[10:0]**: Start of sub-region B
This field defines the address offset of the sub-region B, to be multiplied by the granularity defined in *Table 27*, versus the start of the of memory bank base address.

5.6.11 GTZC1 TZSC register map

Table 32. GTZC1 TZSC register map and reset values

Offset	Register	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
0x000	GTZC1_TZSC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCK
	Reset value																																0
0x004 to 0x00C	Reserved																																
0x010	GTZC1_TZSC_ SECCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2SEC	Res.	Res.	Res.	I2C1SEC	Res.	Res.	Res.	Res.	Res.	IWDGSEC	Res.	Res.	Res.	Res.	Res.	Res.	TIM2SEC
	Reset value															0				0						0							0
0x014	GTZC1_TZSC_ SECCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4SEC	Res.	Res.	Res.	Res.	LPTIM1SEC	I2C3SEC	LPUART1SEC	SPI3SEC	USBSEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1SEC	TIM17SEC	TIM16SEC	Res.	USART1SEC	Res.	Res.	Res.
	Reset value								0					0	0	0	0	0							0	0	0		0			
0x018	GTZC1_TZSC_ SECCFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTACONVSEC	RADIOSEC	RAMCFGSEC	Res.	XSPI1SEC	Res.	Res.	Res.	PKASEC	Res.	Res.	RNGSEC	HASHSEC	AESSEC	Res.	Res.	Res.	Res.	ICACHE_REGSEC	Res.	Res.	CRCSEC	Res.	Res.	Res.
	Reset value								0	0	0		0				0			0	0	0					0			0			
0x01C	Reserved	Reserved																															
0x020	GTZC1_TZSC_ PRIVCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2PRIV	Res.	Res.	Res.	I2C1PRIV	Res.	Res.	Res.	Res.	Res.	IWDGPRIV	Res.	Res.	Res.	Res.	Res.	Res.	TIM2PRIV
	Reset value															0				0						0							0
0x024	GTZC1_TZSC_ PRIVCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4PRIV	Res.	Res.	Res.	Res.	LPTIM1PRIV	I2C3PRIV	LPUART1PRIV	SPI3PRIV	USBPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1PRIV	TIM17PRIV	TIM16PRIV	Res.	USART1PRIV	Res.	Res.	Res.
	Reset value								0					0	0	0	0	0							0	0	0		0			
0x028	GTZC1_TZSC_ PRIVFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTACONVPRIV	RADIOPRIV	RAMCFGPRIV	Res.	XSPI1PRIV	Res.	Res.	Res.	PKAPRIV	Res.	Res.	RNGPRIV	HASHPRIV	AESPRIV	Res.	Res.	Res.	Res.	ICACHE_REGPRIV	Res.	Res.	CRCPRIV	Res.	Res.	Res.
	Reset value								0	0	0		0				0			0	0	0					0			0			
0x02C to 0x03C	Reserved																																
0x040	GTZC1_TZSC_ MPCWM1ACFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0
0x044	GTZC1_TZSC_ MPCWM1AR	Res.	Res.	Res.	Res.	SUBA_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBA_START[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x048	GTZC1_TZSC_ MPCWM1BCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0

Table 32. GTZC1 TZSC register map and reset values (continued)

Offset	Register	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
0x04C	GTZC1_TZSC_ MPCWM1BR	Res.	Res.	Res.	Res.	SUBB_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBB_START[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0

Refer to *Table 26: GTZC sub-blocks address offset*.

5.7 GTZC1 TZIC registers

All registers can only be accessed in words (32 bits).

5.7.1 GTZC1 TZIC interrupt enable register 1 (GTZC1_TZIC_IER1)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged access only.

This register is used to enable interrupt of illegal access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2IE	Res.
														rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	I2C1IE	Res.	Res.	Res.	Res.	Res.	IWDGIE	Res.	Res.	Res.	Res.	Res.	Res.	TIM2IE
		rw						rw							rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **LPTIM2IE**: Illegal access interrupt enable for LPTIM2
- 0: Interrupt disabled
- 1: Interrupt enabled

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **I2C1IE**: Illegal access interrupt enable for I2C1
- 0: Interrupt disabled
- 1: Interrupt enabled

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **IWDGIE**: Illegal access interrupt enable for IWDG
- 0: Interrupt disabled
- 1: Interrupt enabled

Bits 6:1 Reserved, must be kept at reset value.

Bit 0 **TIM2IE**: Illegal access interrupt enable for TIM2
- 0: Interrupt disabled
- 1: Interrupt enabled

5.7.2 GTZC1 TZIC interrupt enable register 2 (GTZC1_TZIC_IER2)

Address offset: 0x004

Reset value: 0x0000 0000

Secure privileged access only.

This register is used to enable interrupt of illegal access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4IE	Res.	Res.	Res.	Res.	LPTIM1IE	I2C3IE	LPUART1IE	SPI3IE
							rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1IE	TIM17IE	TIM16IE	Res.	USART1IE	Res.	Res.	Res.
rw								rw	rw	rw		rw			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **ADC4IE**: Illegal access interrupt enable for ADC4
- 0: Interrupt disabled
- 1: Interrupt enabled

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **LPTIM1IE**: Illegal access interrupt enable for LPTIM1
- 0: Interrupt disabled
- 1: Interrupt enabled

Bit 18 **I2C3IE**: Illegal access interrupt enable for I2C3
- 0: Interrupt disabled
- 1: Interrupt enabled

Bit 17 **LPUART1IE**: Illegal access interrupt enable for LPUART1
- 0: Interrupt disabled
- 1: Interrupt enabled

Bit 16 **SPI3IE**: Illegal access interrupt enable for SPI3
- 0: Interrupt disabled
- 1: Interrupt enabled

Bit 15 **USBIE**: Illegal access interrupt enable for USB
- 0: Interrupt disabled
- 1: Interrupt enabled

Note: This bit is reserved on STM32WBA23xx devices

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **SAI1IE**: Illegal access interrupt enable for SAI1
- 0: Interrupt disabled
- 1: Interrupt enabled

Bit 6 **TIM17IE**: Illegal access interrupt enable for TIM17
- 0: Interrupt disabled
- 1: Interrupt enabled

Bit 5 **TIM16IE**: Illegal access interrupt enable for TIM16
0: Interrupt disabled
1: Interrupt enabled

Bit 4 Reserved, must be kept at reset value.

Bit 3 **USART1IE**: Illegal access interrupt enable for USART1
0: Interrupt disabled
1: Interrupt enabled

Bits 2:0 Reserved, must be kept at reset value.

5.7.3 GTZC1 TZIC interrupt enable register 3 (GTZC1_TZIC_IER3)

Address offset: 0x008

Reset value: 0x0000 0000

Secure privileged access only.

This register is used to enable interrupt of illegal access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTACONVIE	RADIOIE	RAMCFGIE	Res.	XSPI1IE	Res.	Res.	Res.	PKAIE
							rw	rw	rw		rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	RNGIE	HASHIE	AESIE	Res.	Res.	Res.	Res.	ICACHE_REGIE	Res.	Res.	CRCIE	Res.	Res.	Res.
		rw	rw	rw					rw			rw			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **PTACONVIE**: Illegal access interrupt enable for PTACONV
0: Interrupt disabled
1: Interrupt enabled

Bit 23 **RADIOIE**: Illegal access interrupt enable for 2.4 GHz RADIO
0: Interrupt disabled
1: Interrupt enabled

Bit 22 **RAMCFGIE**: Illegal access interrupt enable for RAMCFG
0: Interrupt disabled
1: Interrupt enabled

Bit 21 Reserved, must be kept at reset value.

Bit 20 **XSPI1IE**: Illegal access interrupt enable for XSPI1 and DLYBXS1
0: Interrupt disabled
1: Interrupt enabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **PKAIE**: Illegal access interrupt enable for PKA
0: Interrupt disabled
1: Interrupt enabled

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **RNGIE**: Illegal access interrupt enable for RNG
0: Interrupt disabled
1: Interrupt enabled

Bit 12 **HASHIE**: Illegal access interrupt enable for HASH
0: Interrupt disabled
1: Interrupt enabled

Bit 11 **AESIE**: Illegal access interrupt enable for AES
0: Interrupt disabled
1: Interrupt enabled

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 **ICACHE_REGIE**: Illegal access interrupt enable for ICACHE registers
0: Interrupt disabled
1: Interrupt enabled

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **CRCIE**: Illegal access interrupt enable for CRC
0: Interrupt disabled
1: Interrupt enabled

Bits 2:0 Reserved, must be kept at reset value.

5.7.4 GTZC1 TZIC interrupt enable register 4 (GTZC1_TZIC_IER4)

Address offset: 0x00C

Reset value: 0x0000 0000

Secure privileged access only.

This register is used to enable interrupt of illegal access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MPCBB2IE	SRAM2IE	MPCBB1IE	SRAM1IE	Res.	MPCWM1MEM1IE	Res.	Res.	Res.	Res.
						rw	rw	rw	rw		rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TZICIE	TZSCIE	EXTIIE	Res.	RCCIE	PWRIE	TAMPIE	RTCIE	SYSCFGIE	Res.	Res.	Res.	OTFDEC1IE	FLASH_REGIE	FLASHIE	LPDMA1IE
rw	rw	rw		rw	rw	rw	rw	rw				rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **MPCBB2IE**: Illegal access interrupt enable for MPCBB2
0: Interrupt disabled
1: Interrupt enabled

Bit 24 **SRAM2IE**: Illegal access interrupt enable for SRAM2
0: Interrupt disabled
1: Interrupt enabled

Bit 23 **MPCBB1IE**: Illegal access interrupt enable for MPCBB1
0: Interrupt disabled
1: Interrupt enabled

Bit 22 **SRAM1IE**: Illegal access interrupt enable for SRAM1
0: Interrupt disabled
1: Interrupt enabled

Bit 21 Reserved, must be kept at reset value.

Bit 20 **MPCWM1MEM1IE**: Illegal access interrupt enable for MPCWM1 protected external memory on XSPI1
0: Interrupt disabled
1: Interrupt enabled

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **TZICIE**: Illegal access interrupt enable for GTZC1 TZIC
0: Interrupt disabled
1: Interrupt enabled

Bit 14 **TZSCIE**: Illegal access interrupt enable for GTZC1 TZSC
0: Interrupt disabled
1: Interrupt enabled

Bit 13 **EXTIIE**: Illegal access interrupt enable for EXTI
0: Interrupt disabled
1: Interrupt enabled

Bit 12 Reserved, must be kept at reset value.

Bit 11 **RCCIE**: Illegal access interrupt enable for RCC
0: Interrupt disabled
1: Interrupt enabled

Bit 10 **PWRIE**: Illegal access interrupt enable for PWR
0: Interrupt disabled
1: Interrupt enabled

Bit 9 **TAMPIE**: Illegal access interrupt enable for TAMP
0: Interrupt disabled
1: Interrupt enabled

Bit 8 **RTCIE**: Illegal access interrupt enable for RTC
0: Interrupt disabled
1: Interrupt enabled

Bit 7 **SYSCFGIE**: Illegal access interrupt enable for SYSCFG
0: Interrupt disabled
1: Interrupt enabled

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OTFDEC1IE**: Illegal access interrupt enable for OTFDEC1
0: Interrupt disabled
1: Interrupt enabled

Bit 2 **FLASH_REGIE**: Illegal access interrupt enable for FLASH interface
0: Interrupt disabled
1: Interrupt enabled

Bit 1 **FLASHIE**: Illegal access interrupt enable for FLASH memory
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **LPDMA1IE**: Illegal access interrupt enable for LPDMA1
0: Interrupt disabled
1: Interrupt enabled

5.7.5 GTZC1 TZIC status register 1 (GTZC1_TZIC_SR1)

Address offset: 0x010

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2F	Res.
														r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	I2C1F	Res.	Res.	Res.	Res.	Res.	IWDGF	Res.	Res.	Res.	Res.	Res.	Res.	TIM2F
		r						r							r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **LPTIM2F**: Illegal access flag for LPTIM2
0: No illegal access event
1: Illegal access event

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **I2C1F**: Illegal access flag for I2C1
0: No illegal access event
1: Illegal access event

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **IWDGF**: Illegal access flag for IWDG
0: No illegal access event
1: Illegal access event

Bits 6:1 Reserved, must be kept at reset value.

Bit 0 **TIM2F**: Illegal access flag for TIM2
0: No illegal access event
1: Illegal access event

5.7.6 GTZC1 TZIC status register 2 (GTZC1_TZIC_SR2)

Address offset: 0x014

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4F	Res.	Res.	Res.	Res.	LPTIM1F	I2C3F	LPUART1F	SPI3F
							r					r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USB1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1F	TIM17F	TIM16F	Res.	USART1F	Res.	Res.	Res.
r								r	r	r		r			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **ADC4F**: Illegal access flag for ADC4
- 0: No illegal access event
- 1: Illegal access event

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **LPTIM1F**: Illegal access flag for LPTIM1
- 0: No illegal access event
- 1: Illegal access event

Bit 18 **I2C3F**: Illegal access flag for I2C3
- 0: No illegal access event
- 1: Illegal access event

Bit 17 **LPUART1F**: Illegal access flag for LPUART1
- 0: No illegal access event
- 1: Illegal access event

Bit 16 **SPI3F**: Illegal access flag for SPI3
- 0: No illegal access event
- 1: Illegal access event

Bit 15 **USB1F**: Illegal access flag for USB1
- 0: No illegal access event
- 1: Illegal access event

Note: This bit is reserved on STM32WBA23xx devices

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **SAI1F**: Illegal access flag for SAI1
- 0: No illegal access event
- 1: Illegal access event

Bit 6 **TIM17F**: Illegal access flag for TIM17
- 0: No illegal access event
- 1: Illegal access event

Bit 5 **TIM16F**: Illegal access flag for TIM16
- 0: No illegal access event
- 1: Illegal access event

Bit 4 Reserved, must be kept at reset value.

Bit 3 **USART1F**: Illegal access flag for USART1
0: No illegal access event
1: Illegal access event

Bits 2:0 Reserved, must be kept at reset value.

5.7.7 GTZC1 TZIC status register 3 (GTZC1_TZIC_SR3)

Address offset: 0x018

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTACONVF	RADIOF	RAMCFGF	Res.	XSPI1F	Res.	Res.	Res.	PKAF
							r	r	r		r				r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	RNGF	HASHF	AESF	Res.	Res.	Res.	Res.	ICACHE_REGF	Res.	Res.	CRCF	Res.	Res.	Res.
		r	r	r					r			r			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **PTACONVF**: Illegal access flag for PTACONV
0: No illegal access event
1: Illegal access event

Bit 23 **RADIOF**: Illegal access flag for 2.4 GHz RADIO
0: No illegal access event
1: Illegal access event

Bit 22 **RAMCFGF**: Illegal access flag for RAMCFG
0: No illegal access event
1: Illegal access event

Bit 21 Reserved, must be kept at reset value.

Bit 20 **XSPI1F**: Illegal access flag for XSPI1 and DLYBXS1
0: No illegal access event
1: Illegal access event

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **PKAF**: Illegal access flag for PKA
0: No illegal access event
1: Illegal access event

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **RNGF**: Illegal access flag for RNG
0: No illegal access event
1: Illegal access event

Bit 12 **HASHF**: Illegal access flag for HASH
0: No illegal access event
1: Illegal access event

Bit 11 **AESF**: Illegal access flag for AES
0: No illegal access event
1: Illegal access event

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 **ICACHE_REGF**: Illegal access flag for ICACHE registers
0: No illegal access event
1: Illegal access event

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **CRCF**: Illegal access flag for CRC
0: No illegal access event
1: Illegal access event

Bits 2:0 Reserved, must be kept at reset value.

5.7.8 GTZC1 TZIC status register 4 (GTZC1_TZIC_SR4)

Address offset: 0x01C

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MPCBB2F	SRAM2F	MPCBB1F	SRAM1F	Res.	MPCWM1MEMF	Res.	Res.	Res.	Res.
						r	r	r	r		r				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TZICF	TZSCF	EXTIF	Res.	RCCF	PWRF	TAMPF	RTCF	SYSCFGF	Res.	Res.	Res.	OTFDEC1F	FLASH_REGF	FLASHF	LPDMA1F
r	r	r		r	r	r	r	r				r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **MPCBB2F**: Illegal access flag for MPCBB2
0: No illegal access event
1: Illegal access event

Bit 24 **SRAM2F**: Illegal access flag for SRAM2
0: No illegal access event
1: Illegal access event

Bit 23 **MPCBB1F**: Illegal access flag for MPCBB1
0: No illegal access event
1: Illegal access event

Bit 22 **SRAM1F**: Illegal access flag for SRAM1
0: No illegal access event
1: Illegal access event

Bit 21 Reserved, must be kept at reset value.

Bit 20 **MPCWM1MEMF**: Illegal access flag for MPCWM1 protected external memory
0: No illegal access event
1: Illegal access event

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **TZICF**: Illegal access flag for GTZC1 TZIC
0: No illegal access event
1: Illegal access event

Bit 14 **TZSCF**: Illegal access flag for GTZC1 TZSC
0: No illegal access event
1: Illegal access event

Bit 13 **EXTIF**: Illegal access flag for EXTI
0: No illegal access event
1: Illegal access event

Bit 12 Reserved, must be kept at reset value.

Bit 11 **RCCF**: Illegal access flag for RCC
0: No illegal access event
1: Illegal access event

Bit 10 **PWRF**: Illegal access flag for PWR
0: No illegal access event
1: Illegal access event

Bit 9 **TAMPF**: Illegal access flag for TAMP
0: No illegal access event
1: Illegal access event

Bit 8 **RTCF**: Illegal access flag for RTC
0: No illegal access event
1: Illegal access event

Bit 7 **SYSCFGF**: Illegal access flag for SYSCFG
0: No illegal access event
1: Illegal access event

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OTFDEC1F**: Illegal access flag for OTFDEC1
0: No illegal access event
1: Illegal access event

Bit 2 **FLASH_REGF**: Illegal access flag for FLASH interface
0: No illegal access event
1: Illegal access event

Bit 1 **FLASHF**: Illegal access flag for FLASH memory
0: No illegal access event
1: Illegal access event

Bit 0 **LPDMA1F**: Illegal access flag for LPDMA1
0: No illegal access event
1: Illegal access event

5.7.9 GTZC1 TZIC flag clear register 1 (GTZC1_TZIC_FCR1)

Address offset: 0x020

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLPTIM2F	Res.
														w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CI2C1F	Res.	Res.	Res.	Res.	Res.	CIWDGF	Res.	Res.	Res.	Res.	Res.	Res.	CTIM2F
		w						w							w

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **CLPTIM2F**: Clear the illegal access flag for LPTIM2
0: No action
1: Status flag cleared

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **CI2C1F**: Clear the illegal access flag for I2C1
0: No action
1: Status flag cleared

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **CIWDGF**: Clear the illegal access flag for IWDG
0: No action
1: Status flag cleared

Bits 6:1 Reserved, must be kept at reset value.

Bit 0 **CTIM2F**: Clear the illegal access flag for TIM2
0: No action
1: Status flag cleared

5.7.10 GTZC1 TZIC flag clear register 2 (GTZC1_TZIC_FCR2)

Address offset: 0x024

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	CADC4F	Res.	Res.	Res.	Res.	CLPTIM1F	CI2C3F	CLPUART1F	CSPI3F
							w					w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CUSBF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSAI1F	CTIM17F	CTIM16F	Res.	CUSART1F	Res.	Res.	Res.
w								w	w	w		w			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **CADC4F**: Clear the illegal access flag for ADC4
0: No action
1: Status flag cleared

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **CLPTIM1F**: Clear the illegal access flag for LPTIM1
0: No action
1: Status flag cleared

Bit 18 **CI2C3F**: Clear the illegal access flag for I2C3
0: No action
1: Status flag cleared

Bit 17 **CLPUART1F**: Clear the illegal access flag for LPUART1
0: No action
1: Status flag cleared

Bit 16 **CSPI3F**: Clear the illegal access flag for SPI3
0: No action
1: Status flag cleared

Bit 15 **CUSBF**: Clear the illegal access flag for USB
0: No action
1: Status flag cleared
Note: This bit is reserved on STM32WBA23xx devices

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **CSAI1F**: Clear the illegal access flag for SAI1
0: No action
1: Status flag cleared

Bit 6 **CTIM17F**: Clear the illegal access flag for TIM17
0: No action
1: Status flag cleared

Bit 5 **CTIM16F**: Clear the illegal access flag for TIM16
0: No action
1: Status flag cleared

Bit 4 Reserved, must be kept at reset value.

Bit 3 **CUSART1F**: Clear the illegal access flag for USART1
0: No action
1: Status flag cleared

Bits 2:0 Reserved, must be kept at reset value.

5.7.11 GTZC1 TZIC flag clear register 3 (GTZC1_TZIC_FCR3)

Address offset: 0x028

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTACONVF	CRADIOF	CRAMCFGF	Res.	CXSPI1F	Res.	Res.	Res.	CPKAF
							w	w	w		w				w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CRNGF	CHASHF	CAESF	Res.	Res.	Res.	Res.	CICACHE_REGF	Res.	Res.	CCRCF	Res.	Res.	Res.
		w	w	w					w			w			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **CPTACONVF**: Clear the illegal access flag for PTACONV
- 0: No action
- 1: Status flag cleared

Bit 23 **CRADIOF**: Clear the illegal access flag for 2.4 GHz RADIO
- 0: No action
- 1: Status flag cleared

Bit 22 **CRAMCFGF**: Clear the illegal access flag for RAMCFG
- 0: No action
- 1: Status flag cleared

Bit 21 Reserved, must be kept at reset value.

Bit 20 **CXSPI1F**: Clear the illegal access flag for XSPI1 and DLYBXS1
- 0: No action
- 1: Status flag cleared

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **CPKAF**: Clear the illegal access flag for PKA
- 0: No action
- 1: Status flag cleared

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **CRNGF**: Clear the illegal access flag for RNG
- 0: No action
- 1: Status flag cleared

Bit 12 **CHASHF**: Clear the illegal access flag for HASH
- 0: No action
- 1: Status flag cleared

Bit 11 **CAESF**: Clear the illegal access flag for AES
- 0: No action
- 1: Status flag cleared

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 **CICACHE_REGF**: Clear the illegal access flag for ICACHE registers
0: No action
1: Status flag cleared

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **CCRCF**: Clear the illegal access flag for CRC
0: No action
1: Status flag cleared

Bits 2:0 Reserved, must be kept at reset value.

5.7.12 GTZC1 TZIC flag clear register 4 (GTZC1_TZIC_FCR4)

Address offset: 0x02C

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CMPCBB2F	CSRAM2F	CMPCBB1F	CSRAM1F	Res.	CMPCWM1MEM1F	Res.	Res.	Res.	Res.
						w	w	w	w		w				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTZICF	CTZSCF	CEXTIF	Res.	CRCCF	CPWRF	CTAMPF	CRTCF	CSYSCFGF	Res.	Res.	Res.	COTFDEC1F	CFLASH_REGF	CFLASHF	CLPDMA1F
w	w	w		w	w	w	w	w				w	w	w	w

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **CMPCBB2F**: Clear the illegal access flag for MPCBB2
0: No action
1: Status flag cleared

Bit 24 **CSRAM2F**: Clear the illegal access flag for SRAM2
0: No action
1: Status flag cleared

Bit 23 **CMPCBB1F**: Clear the illegal access flag for MPCBB1
0: No action
1: Status flag cleared

Bit 22 **CSRAM1F**: Clear the illegal access flag for SRAM1
0: No action
1: Status flag cleared

Bit 21 Reserved, must be kept at reset value.

Bit 20 **CMPCWM1MEM1F**: Clear the illegal access flag for MPCWM1 protected external memory
0: No action
1: Status flag cleared

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **CTZICF**: Clear the illegal access flag for GTZC1 TZIC
0: No action
1: Status flag cleared

Bit 14 **CTZSCF**: Clear the illegal access flag for GTZC1 TZSC
0: No action
1: Status flag cleared

Bit 13 **CEXTIF**: Clear the illegal access flag for EXTI
0: No action
1: Status flag cleared

Bit 12 Reserved, must be kept at reset value.

Bit 11 **CRCCF**: Clear the illegal access flag for RCC
0: No action
1: Status flag cleared

Bit 10 **CPWRF**: Clear the illegal access flag for PWR
0: No action
1: Status flag cleared

Bit 9 **CTAMPF**: Clear the illegal access flag for TAMP
0: No action
1: Status flag cleared

Bit 8 **CRTCF**: Clear the illegal access flag for RTC
0: No action
1: Status flag cleared

Bit 7 **CSYSCFGF**: Clear the illegal access flag for SYSCFG
0: No action
1: Status flag cleared

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **COTFDEC1F**: Clear the illegal access flag for OTFDEC1
0: No action
1: Status flag cleared

Bit 2 **CFLASH_REGF**: Clear the illegal access flag for FLASH interface
0: No action
1: Status flag cleared

Bit 1 **CFLASHF**: Clear the illegal access flag for FLASH memory
0: No action
1: Status flag cleared

Bit 0 **CLPDMA1F**: Clear the illegal access flag for LPDMA1
0: No action
1: Status flag cleared

5.7.13 GTZC1 TZIC register map

Table 33. GTZC1 TZIC register map and reset values

Offset	Register	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
0x000	GTZC1_TZIC_IER1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2IE	Res.	Res.	Res.	I2C1IE	Res.	Res.	Res.	Res.	Res.	IWDGIE	Res.	Res.	Res.	Res.	Res.	Res.	TIM2IE
	Reset value															0				0						0							0
0x004	GTZC1_TZIC_IER2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4IE	Res.	Res.	Res.	Res.	LPTIM1IE	I2C3IE	LPUART1IE	SPI3IE	USBIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1IE	TIM17IE	TIM16IE	Res.	USART1IE	Res.	Res.	Res.	Res.
	Reset value								0					0	0	0	0	0								0	0	0		0				
0x008	GTZC1_TZIC_IER3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTACONVIE	RADIOIE	RAMCFGIE	Res.	XSPI1IE	Res.	Res.	Res.	PKAIE	Res.	Res.	RNGIE	HASHIE	AESIE	Res.	Res.	Res.	Res.	ICACHE_REGIE	Res.	Res.	Res.	CRCIE	Res.	Res.	Res.
	Reset value								0	0	0		0				0			0	0	0				0					0			
0x00C	GTZC1_TZIC_IER4	Res.	Res.	Res.	Res.	Res.	Res.	MPCBB2IE	SRAM2IE	MPCBB1IE	SRAM1IE	Res.	MPCWM1MEMIE	Res.	Res.	Res.	Res.	TZICIE	TZSCIE	EXTIIE	Res.	RCCIE	PWRIE	TAMPIE	RTCIE	SYSCFGIE	Res.	Res.	Res.	OTFDEC1IE	FLASH_REGIE	FLASHIE	LPDMA1IE
	Reset value							0	0	0	0		0					0	0	0		0	0	0	0	0				0	0	0	0
0x010	GTZC1_TZIC_SR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2F	Res.	Res.	Res.	I2C1F	Res.	Res.	Res.	Res.	Res.	IWDGF	Res.	Res.	Res.	Res.	Res.	Res.	TIM2F
	Reset value															0				0						0							0
0x014	GTZC1_TZIC_SR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4F	Res.	Res.	Res.	Res.	LPTIM1F	I2C3F	LPUART1F	SPI3F	USBF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1F	TIM17F	TIM16F	Res.	USART1F	Res.	Res.	Res.	Res.
	Reset value								0					0	0	0	0	0								0	0	0		0				
0x018	GTZC1_TZIC_SR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTACONVF	RADIOF	RAMCFGF	Res.	XSPI1F	Res.	Res.	Res.	PKAF	Res.	Res.	RNGF	HASHF	AESF	Res.	Res.	Res.	Res.	ICACHE_REGF	Res.	Res.	CRCF	Res.	Res.	Res.	Res.
	Reset value								0	0	0		0				0			0	0	0				0								
0x01C	GTZC1_TZIC_SR4	Res.	Res.	Res.	Res.	Res.	Res.	MPCBB2F	SRAM2F	MPCBB1F	SRAM1F	Res.	MPCWM1MEM1F	Res.	Res.	Res.	Res.	TZICF	TZSCF	EXTIF	Res.	RCCF	PWRF	TAMPF	RTCF	SYSCFGF	Res.	Res.	Res.	OTFDEC1F	FLASH_REGF	FLASHF	LPDMA1F
	Reset value							0	0	0	0		0					0	0	0		0	0	0	0	0				0	0	0	0
0x020	GTZC1_TZIC_ FCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLPTIM2F	Res.	Res.	Res.	CI2C1F	Res.	Res.	Res.	Res.	Res.	CIWDGF	CWWDGF	Res.	Res.	Res.	Res.	Res.	CTIM2F
	Reset value															0				0						0	0						0
0x024	GTZC1_TZIC_ FCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CADC4F	Res.	Res.	Res.	Res.	CLPTIM1F	CI2C3F	CLPUART1F	CSPI3F	CUSBF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSAI1F	CTIM17F	CTIM16F	Res.	CUSART1F	Res.	Res.	Res.	Res.
	Reset value								0					0	0	0	0	0								0	0	0		0				

Table 33. GTZC1 TZIC register map and reset values (continued)

Offset	Register	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
0x028	GTZC1_TZIC_FCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTACONVF	CRADIOF	CRAMCFGF	Res.	CXSPI1F	Res.	Res.	Res.	CPKAF	Res.	Res.	CRNGF	CHASHF	CAESF	Res.	Res.	Res.	Res.	CICACHE_REGF	Res.	Res.	CCRCF	Res.	Res.	Res.
	Reset value								0	0	0		0				0			0	0	0					0			0			
0x02C	GTZC1_TZIC_FCR4	Res.	Res.	Res.	Res.	Res.	Res.	CMPCBB2F	CSRAM2F	CMPCBB1F	CSRAM1F	Res.	CMPCWM1MEMF	Res.	Res.	Res.	Res.	CTZICF	CTZSCF	CEXTIF	Res.	CRCCF	CPWRF	CTAMPF	CRTCF	CSYSCFGF	Res.	Res.	Res.	OTFDEC1F	CFLASH_REGF	CFLASHF	CLPDMA1F
	Reset value							0	0	0	0		0					0	0	0		0	0	0	0	0				0	0	0	0

Refer to *Table 26: GTZC sub-blocks address offset*.

5.8 GTZC1 MPCBB registers

All registers can only be accessed in words (32 bits).

5.8.1 GTZC1 MPCBB control register (GTZC1_MPCBBx_CR) (x = 1 to 2)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRWILADIS	INVSECSTATE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
															rs

Bit 31 **SRWILADIS**: secure read/write illegal access disable

This bit disables the detection of an illegal access when a secure read/write transaction access a nonsecure blocks of the block-based SRAMx (secure fetch on nonsecure block is always considered illegal).

0: enabled, secure read/write acces not allowed on nonsecure SRAMx block

1: disabled, secure read/write access allowed on nonsecure SRAMx block

Bit 30 **INVSECSTATE**: SRAM clocks security state

This bit is used to define the internal SRAM clocks control in RCC as secure or not.

0: SRAM clock is secured if a secure area exists in the MPCBBx. It is nonsecure if there is no secure area.

1: SRAM clock is nonsecure even if a secure area exists in the MPCBBx, and secure even if no secure block is set in the MPCBBx.

Bits 29:1 Reserved, must be kept at reset value.

Bit 0 **GLOCK**: Lock the control register of the MPCBBx until next reset
This bit is cleared by default and once set, it can not be reset until system reset.
0: Control register not locked
1: Control register locked

5.8.2 GTZC1 MPCBB1 configuration lock register (GTZC1_MPCBB1_CFGLOCK)

Address offset: 0x010

Reset value: 0x0000 0000

Secure privileged access only.

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **SPLCK[3:0]**: Security/privilege configuration lock super-block (n = 0 to 3)
This bit is set by software and can be cleared only by system reset.
0: GTZC1_MPCBB1_SECCFGRn and GTZC1_MPCBB1_PRIVCFGRn can be written.
1: Writes to GTZC1_MPCBB1_SECCFGRn and GTZC1_MPCBB1_PRIVCFGRn are ignored

5.8.3 GTZC1 MPCBB2 configuration lock register (GTZC1_MPCBB2_CFGLOCK)

Address offset: 0x010

Reset value: 0x0000 0000

Secure privileged access only.

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

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **SPLCK[1:0]**: Security/privilege configuration lock super-block (n = 0 to 1)
This bit is set by software and can be cleared only by system reset.
0: GTZC1_MPCBB2_SECCFGRn and GTZC1_MPCBB2_PRIVCFGRn can be written.
1: Writes to GTZC1_MPCBB2_SECCFGRn and GTZC1_MPCBB2_PRIVCFGRn are ignored

5.8.4 GTZC1 MPCBB security configuration for superblock n register (GTZC1_MPCBBx_SECCFGRn) (x = 1 to 2)

Address offset: 0x100 + 0x04 * n, (n = 0 to 1)

Reset value: 0xFFFF FFFF

The given reset value is valid when TZEN = 1. The reset value is 0x0000 0000 when TZEN = 0.

Write access to this register is secure only. Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SEC[31:0]**: Security configuration for block y (y = 0 to 31) in super block n
0: Nonsecure access only to block y, belonging to super-block n. Secure access is also allowed if the SRWILADIS bit is set in GTZC1_MPCBBx_CR.
1: Secure access only to block y, belonging to super-block n.
Unprivileged write to this bit is ignored if PRIVy bit is set in GTZC1_MPCBBx_PRIVCFGRn.
Writes are ignored if SPLCKn bit is set in GTZC1_MPCBBx_CFGLOCK.

5.8.5 GTZC1 MPCBB1 security configuration for superblock n register (GTZC1_MPCBB1_SECCFGRn)

Address offset: 0x100 + 0x04 * n, (n = 2 to 3)

Reset value: 0xFFFF FFFF

The given reset value is valid when TZEN = 1. The reset value is 0x0000 0000 when TZEN = 0.

Write access to this register is secure only. Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SEC[31:0]**: Security configuration for block y (y = 0 to 31) in super block n
0: Nonsecure access only to block y, belonging to super-block n. Secure access is also allowed if the SRWILADIS bit is set in GTZC1_MPCBB1_CR.
1: Secure access only to block y, belonging to super-block n.
Unprivileged write to this bit is ignored if PRIVy bit is set in GTZC1_MPCBB1_PRIVCFGRn.
Writes are ignored if SPLCKn bit is set in GTZC1_MPCBB1_CFGLOCK.

5.8.6 GTZC1 MPCBB privileged configuration for superblock n register (GTZC1_MPCBBx_PRIVCFGRn) (x = 1 to 2)

Address offset: 0x200 + 0x04 * n, (n = 0 to 1)

Reset value: 0xFFFF FFFF

The given reset value is valid when TZEN = 1. The reset value is 0x0000 0000 when TZEN = 0.

Write access to this register is privileged only. Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIV[31:0]**: Privileged configuration for block y (y = 0 to 31), belonging to super-block n.
0: Privileged and unprivileged access to block y, belonging to super-block n
1: Only privileged access to block y, belonging to super-block n
Nonsecure write to this bit is ignored if SECy bit is set in GTZC1_MPCBBx_SECCFGRn.
Writes are ignored if SPLCKn bit is set in GTZC1_MPCBBx_CFGLOCK.

5.8.7 GTZC1 MPCBB1 privileged configuration for superblock n register (GTZC1_MPCBB1_PRIVCFGRn)

Address offset: 0x200 + 0x04 * n, (n = 2 to 3)

Reset value: 0xFFFF FFFF

The given reset value is valid when TZEN = 1. The reset value is 0x0000 0000 when TZEN = 0.

Write access to this register is privileged only. Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIV[31:0]**: Privileged configuration for block y (y = 0 to 31), belonging to super-block n.
0: Privileged and unprivileged access to block y, belonging to super-block n
1: Only privileged access to block y, belonging to super-block n
Nonsecure write to this bit is ignored if SECy bit is set in GTZC1_MPCBB1_SECCFGRn.
Writes are ignored if SPLCKn bit is set in GTZC1_MPCBB1_CFGLOCK.

5.8.8 GTZC1 MPCBB1 register map

Table 34. GTZC1 MPCBB1 register map and reset values

Offset	Register	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
0x000	GTZC1_ MPCBB1_CR	SRWILADIS	INVSECSTATE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
	Reset value	0	0																														0
0x004 to 0x00C	Reserved																																
0x010	GTZC1_MPCBB1_ CFGLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPLCK3	SPLCK2	SPLCK1	SPLCK0
	Reset value																													0	0	0	0
0x014 to 0x0FC	Reserved																																
0x100	GTZC1_MPCBB1_ SECCFGR0	SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x104	GTZC1_MPCBB1_ SECCFGR1	SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x108)	GTZC1_MPCBB1_ SECCFGR2	SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x10C	GTZC1_MPCBB1_ SECCFGR3	SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x110 to 0x1FC	Reserved																																
0x200	GTZC1_MPCBB1_ PRIVCFGR0	PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x204	GTZC1_MPCBB1_ PRIVCFGR1	PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x208	GTZC1_MPCBB1_ PRIVCFGR2	PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x20C	GTZC1_MPCBB1_ PRIVCFGR3	PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x210 to 0x3FC	Reserved																																

Refer to *Table 26: GTZC sub-blocks address offset*.

Table 35. GTZC1 MPCBB2 register map and reset values

Offset	Register	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
0x000	GTZC1_ MPCBB2_CR	SRWILADIS	INVSECSTATE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
	Reset value	0	0																														0
0x004 to 0x00C	Reserved																																
0x010	GTZC1_MPCBB2_ CFGLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPLCK1	SPLCK0
	Reset value																															0	0
0x014 to 0x0FC	Reserved																																
0x100	GTZC1_MPCBB2_ SECCFGR0	SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x104	GTZC1_MPCBB2_ SECCFGR1	SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x108 to 0x1FC	Reserved																																
0x200	GTZC1_MPCBB2_ PRIVCFGR0	PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x204	GTZC1_MPCBB2_ PRIVCFGR1	PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x208 to 0x3FC	Reserved																																

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

6 RAMs configuration controller (RAMCFG)

6.1 Introduction

The RAMCFG configures the features of the internal SRAM (SRAM1 and SRAM2).

6.2 RAMCFG main features

The internal SRAM supports the following features, configured in RAMCFG:

- Parity error detection with failing address indication
- Write protection (1-Kbyte granularity)
- Programmable wait states
- SRAM software erase

6.3 RAMCFG functional description

6.3.1 Internal SRAM features

The main SRAM (composed of SRAM1 and SRAM2) is embedded in the device, with 2.4 GHz RADIO RXTXRAM and sequence SRAM, each with specific features:

- SRAM power/retention control
  - SRAM1 and/or SRAM2 powered down in Stop modes to reduce consumption
  - SRAM1, SRAM2, and/or 2.4 GHz RADIO SRAMs retained in Standby mode

  Refer to *Section 11: Power control (PWR)* for more details.
- SRAM erase on RDP regression:
  - The internal SRAM1 and SRAM2 are erased by hardware in case of readout protection (RDP) level regression to Level 0.

  Refer to *Section 7: Embedded flash memory (FLASH)* for more details.
- SRAM erase on reset:
  - SRAM1 is erased when a system reset occurs if the SRAM1_RST option bit is selected in the flash memory user option bytes.
  - SRAM2 is erased when a system reset occurs if the SRAM2_RST option bit is selected in the flash memory user option bytes.

  Refer to *Section 7: Embedded flash memory (FLASH)* for more details.
- Tamper protection
  - SRAM2 is protected by the tamper detection circuit and can be erased by hardware in case of tamper detection.
  - It is also erased by hardware in case of a V_{DD} BOR0 (Backup domain supply power-on).

  Refer to *Section 34: Tamper and backup registers (TAMP)* for more details.

- Parity and write protection:
  - SRAM2 features parity and write protection.

  Refer to *Section 6.3.2: Internal SRAM parity* and *Section 6.3.2: Internal SRAM parity* for more details.
- Wait states and software reset:
  - SRAM1 and SRAM2 feature programmable wait state configuration.
  - SRAM1 and SRAM2 feature software erase.

  Refer to *Section 6.3.2* and *Section 6.3.5* for more details.

Table 36 summarizes the features supported by each internal SRAM.

Table 36. Internal SRAMs features

Feature	SRAM1	SRAM2	2.4 GHz RADIO SRAMs
LPBAM in Stop 0 mode	X	X	-
Optional retention in Stop mode	X	X	-(1)
Optional retention in Standby mode	X	X	X
Erased with RDP regression	X	X	-
Erased/blocked with tamper detection	-	X	-
Optionally erased with system reset	X	X	-
Software erase	X	X	-
Parity	-	X	-
Write protection	-	X	-
Wait states	X	X	-

1. 2.4 GHz RADIO SRAMs are always retained in Stop modes.

6.3.2 Internal SRAM parity

Parity is supported by SRAM2 when enabled with the SRAM2_PE user option bit and when either PEIE or PENMI is set. Refer to *Section 7: Embedded flash memory (FLASH)* for more details.

Four parity bits are added for every 32 bits of SRAM (one bit per byte) to increase memory robustness.

The parity bits are computed and stored when writing in SRAM2, and are automatically checked when reading. If one byte parity fails, an NMI or IRQ can be generated. The same error can also be linked to the break input (BRK_IN) of TIM16 or TIM17 with the SYSCFG_CFGR2.SPL control bit.

Note: When enabling SRAM2 parity, it is advised to software-initialize the entire RAM memory at the beginning of the code. This avoids parity errors when reading noninitialized addresses.

When a parity error is detected, the PED bit is set in the *RAMCFG SRAM2 interrupt status register (RAMCFG_M2ISR)*. An interrupt or NMI can be generated if enabled by the PEIE or PENMI bits in the *RAMCFG SRAM2 interrupt enable register (RAMCFG_M2IER)*. When the ALE bit is set in the *RAMCFG SRAM2 control register (RAMCFG_M2CR)*, the SRAM offset word address at which the parity error is detected is stored in the *RAMCFG SRAM2 parity*

error address register (RAMCFG_M2PEAR), alongside the parity error flag of the failing byte(s) and the AHB bus master ID.

If the PED bit is already set when a second parity error occurs, the SRAM offset word address, the failing byte(s) parity error flag, and the AHB bus master ID are not updated. These are only updated when PED is cleared.

Unaligned word or half-word access may generate a parity error on bytes not belonging to the accessed data. The table below provides more details.

Table 37. SRAM access parity errors

Access type	Error report
Aligned	All reported errors belong to the bytes read.
Unaligned word at byte address offset 0x1	An error on word offset address byte 0 does not belong to the read data.
Unaligned word at byte address offset 0x2	All reported errors belong to the bytes read.
Unaligned word at byte address offset 0x3	An error on second word offset address byte 3 does not belong to the read data.
Unaligned half-word at byte address offset 0x1	An error on word offset address byte 0 or 3 does not belong to the read data.
Unaligned half-word at byte address offset 0x3	All reported errors belong to the bytes read.

Table 38. SRAM parity error bus master ID

	CPU	Debugger	DMA
SRAM2	0b010	0b011	0b110

6.3.3 Internal SRAM write protection

The SRAM2 is made up of 32 1-Kbyte write-protect pages. Each 1-Kbyte write-protect page can be write-protected by setting its corresponding PyWP (y = 0 to 31) bit in the *RAMCFG SRAM2 write protection register 1 (RAMCFG_M2WPR1)*.

A write operation to a write-protected page is ignored and a bus error is generated.

Any SRAM erase operation also erases any write-protected pages.

6.3.4 Internal SRAM read access latency

To read data from the SRAM correctly, the number of wait states (WS) must be programmed in the WSC[2:0] bitfield of the *RAMCFG SRAM1 control register (RAMCFG_M1CR)*. This is dependent on the voltage scaling range and the AHB hclk1 clock frequency (see *Table 39*).

Table 39. Number of wait states versus hclk frequency and voltage range scaling

Wait states (latency)	AHB hclk1 (MHz)		
	V_{CORE} range 1	V_{CORE} range 1.5	V_{CORE} range 2
0 WS (1 AHB cycle)	≤ 64	≤ 64	≤ 12
1 WS (2 AHB cycles)	-	-	≤ 16

After reset, the SRAM1 and SRAM2 hclk1 frequency is 16 MHz and one wait state is configured in the WSC bitfield of the RAMCFG_MxCR register.

Caution: Before entering Stop 1, 2, and 3 modes, software must set the SRAM1 and SRAM2 wait states to at least 1 in the RAMCFG_MxCR register to comply with the SYSCLK 16 MHz and range 2 configuration when exiting Stop 1, 2, and 3 modes.

6.3.5 Internal SRAM erase

An SRAM erase can be requested by executing this software sequence:

1. Write 0xCA in the *RAMCFG SRAM erase key register (RAMCFG_MxERKEYR)*.
2. Write 0x53 in the *RAMCFG SRAM erase key register (RAMCFG_MxERKEYR)*.
3. Set the SRAMER bit of the *RAMCFG SRAM1 control register (RAMCFG_M1CR)*.

An SRAM erase can also be started upon a reset condition (see *Section 7.4: FLASH option bytes*).

The SRAMBUSY flag is set in the related SRAM interrupt status register (*RAMCFG SRAM1 interrupt status register (RAMCFG_M1ISR)* or *RAMCFG SRAM2 interrupt status register (RAMCFG_M2ISR)*) for the entire duration of the erase operation.

The total duration of each SRAM erase operation is N x AHB clock cycles, where N is the size of the SRAM in 32-bit words.

If the SRAM is read while an erase operation is ongoing, a default zero data is read on the AHB bus.

If the SRAM is written while an erase operation is ongoing, wait states are inserted on the AHB bus until the end of the erase operation.

6.4 RAMCFG low-power modes

Table 40. Effect of low-power modes on RAMCFG

Mode	Description
Sleep	No effect. RAMCFG interrupts cause the device to exit Sleep mode.
Stop	The content of RAMCFG registers is kept. The SRAM2 parity is functional and an interrupt or NMI causes the device to exit Stop 0 mode.
Standby	The RAMCFG peripheral is powered down and must be reinitialized after exiting Standby.

6.5 RAMCFG interrupts

Table 41. RAMCFG interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop mode	Exit Standby mode
RAMCFG	Parity error detection	PED	PEIE and PENMI = 0	Set CPED	Yes	Yes(1)	No
NMI	Parity error detection	PED	PENMI	Set CPED	Yes	Yes(1)	No

1. Stop 0 mode only

6.6 RAMCFG registers

Access to all RAMCFG registers can be protected by GTZC1_TZSC RAMCFGSEC and GTZC1_TZSC RAMCFGPRIV.

6.6.1 RAMCFG SRAM1 control register (RAMCFG_M1CR)

Address offset: 0x000

Reset value: 0x0001 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WSC[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rs								

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **WSC[2:0]**: SRAM1 wait state configuration

This field is used to program the number of wait states inserted on the AHB when reading the SRAM1, depending on its access time.

000: Zero wait states

001: One wait state

...

111: Seven wait states

Note: Before entering Stop 1, 2, and 3 mode software must set SRAM1 wait states to at least 1.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **SRAMER**: SRAM1 erase

This bit can be set by software only after writing the unlock sequence in the ERASEKEY field of the *RAMCFG SRAM erase key register (RAMCFG_MxERKEYR)*. Setting this bit starts the SRAM1 erase. This bit is automatically cleared by hardware at the end of the erase operation.

0: No erase operation ongoing

1: Erase operation ongoing

Bits 7:0 Reserved, must be kept at reset value.

6.6.2 RAMCFG SRAM1 interrupt status register (RAMCFG_M1ISR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMBUSY**: SRAM busy with erase operation.

0: No memory erase operation ongoing

1: Memory erase operation ongoing

Note: Depending on the SRAM, the erase operation can be performed due to software request, system reset if the enabled by user option, tamper detection or RDP regression. Refer to Table 36.

Bits 7:0 Reserved, must be kept at reset value.

6.6.3 RAMCFG SRAM erase key register (RAMCFG_MxERKEYR)

Address offset: 0x028 + 0x040 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **ERASEKEY[7:0]**: Erase write protection key

The following steps are required to unlock the write protection of the SRAMER bit in the *RAMCFG SRAM1 control register (RAMCFG_M1CR)* or the *RAMCFG SRAM2 control register (RAMCFG_M2CR)*.

a) Write 0xCA into ERASEKEY[7:0]

b) Write 0x53 into ERASEKEY[7:0]

Note: Writing a wrong key reactivates the write protection.

6.6.4 RAMCFG SRAM2 control register (RAMCFG_M2CR)

Address offset: 0x040

Reset value: 0x0001 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WSC[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	Res.
							rs				rw				

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **WSC[2:0]**: SRAM2 wait state configuration

This field is used to program the number of wait states inserted on the AHB when reading the SRAM2, depending on its access time.

000: 0 wait state

001: 1 wait state

...

111: 7 wait states

Note: Before entering Stop 1, 2, and 3 modes, software shall set SRAM2 wait states to at least 1.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **SRAMER**: SRAM2 erase

This bit can be set by software only after writing the unlock sequence in the ERASEKEY field of the *RAMCFG SRAM erase key register (RAMCFG_MxERKEYR)*. Setting this bit starts the SRAM2 erase. This bit is automatically cleared by hardware at the end of the erase operation.

0: No erase operation ongoing

1: Erase operation ongoing

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **ALE**: SRAM2 parity fail address latch enable

0: Failing address not stored in the *RAMCFG SRAM2 parity error address register (RAMCFG_M2PEAR)*

1: Failing address stored in the *RAMCFG SRAM2 parity error address register (RAMCFG_M2PEAR)*

Bits 3:0 Reserved, must be kept at reset value.

6.6.5 RAMCFG SRAM2 interrupt enable register (RAMCFG_M2IER)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PENMI	Res.	PEIE	Res.
												rs		rw	

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **PENMI**: Parity error NMI.

This bit is set by software and cleared only by a global RAMCFG reset.

0: NMI not generated in case of a parity error

1: NMI generated in case of a parity error

Note: When the PENMI bit is set, the RAMCFG maskable interrupt is not generated for a parity error, whatever the PEIE bit value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **PEIE**: Parity error interrupt enable

0: Parity error interrupt disabled

1: Party error interrupt enabled

Bit 0 Reserved, must be kept at reset value.

6.6.6 RAMCFG SRAM2 interrupt status register (RAMCFG_M2ISR)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	PED	Res.
							r							r	

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMBUSY**: SRAM2 busy with erase operation.

0: No memory erase operation ongoing

1: Memory erase operation ongoing

Note: Depending on the SRAM2, the erase operation can be performed through a software request, system reset if enabled by user option, tamper detection, or RDP regression. Refer to Table 36: Internal SRAMs features.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **PED**: Parity error detected

0: No parity error detected

1: Parity error detected

Bit 0 Reserved, must be kept at reset value.

6.6.7 RAMCFG SRAM2 parity error address register (RAMCFG_M2PEAR)

Address offset: 0x050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BYTE[3:0]				ID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **BYTE[3:0]**: Byte parity error flag.

When the ALE bit is set in the *RAMCFG SRAM2 control register (RAMCFG_M2CR)*, this field is updated when PED is cleared and a new parity error is detected, with:

1xxx: Parity error detected on the fourth byte in word-aligned address

x1xx: Parity error detected on the third byte in word-aligned address

xx1x: Parity error detected on the second byte in word-aligned address

xxx1: Parity error detected on the first byte in word-aligned address

Bits 27:24 **ID[3:0]**: Parity error AHB bus master ID.

When the ALE bit is set in the *RAMCFG SRAM2 control register (RAMCFG_M2CR)* register, this field is updated when PED is cleared and a new parity error is detected, with:

010: Parity error detected on CPU access

011: Parity error detected on Debugger access

110: Parity error detected on DMA master port 0 access

Others: reserved

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:0 **PEA[15:0]**: Parity error SRAM word aligned address offset.

PEA[1:0] read always 0b00.

When the ALE bit is set in the *RAMCFG SRAM2 control register (RAMCFG_M2CR)* register, this field is updated when PED is cleared and a new parity error is detected, with the SRAM word aligned address offset corresponding to the parity error.

6.6.8 RAMCFG SRAM2 interrupt clear register (RAMCFG_M2ICR)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPED	Res.
														rc_w1	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CPED**: Clear parity error detect bit

Setting this bit clears the PED bit in the *RAMCFG SRAM2 interrupt status register (RAMCFG_M2ISR)*.

Reading this bit returns the value of the *RAMCFG SRAM2 interrupt status register (RAMCFG_M2ISR)* PED bit.

Bit 0 Reserved, must be kept at reset value.

6.6.9 RAMCFG SRAM2 write protection register 1 (RAMCFG_M2WPR1)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P31WP	P30WP	P29WP	P28WP	P27WP	P26WP	P25WP	P24WP	P23WP	P22WP	P21WP	P20WP	P19WP	P18WP	P17WP	P16WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15WP	P14WP	P13WP	P12WP	P11WP	P10WP	P9WP	P8WP	P7WP	P6WP	P5WP	P4WP	P3WP	P2WP	P1WP	P0WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **PyWP**: SRAM2 1-Kbyte write-protect page y write protection (y = 31 to 0)

These bits are set by software and cleared only by a system reset.

0: Write protection of SRAM2 1-Kbyte write protect page y is disabled.

1: Write protection of SRAM2 1-Kbyte write protect page y is enabled.

6.6.10 RAMCFG register map

Table 42. RAMCFG register map and reset values

Offset	Register name reset	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
0x000	RAMCFG_ M1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WSC[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	1								0								
0x004	Reserved	Reserved																															
0x008	RAMCFG_ M1ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x00C to 0x024	Reserved	Reserved																															
0x028	RAMCFG_ M1ERKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x02C to 0x03C	Reserved	Reserved																															
0x040	RAMCFG_ M2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WSC[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	Res.
	Reset value														0	0	1								0				0				
0x044	RAMCFG_ M2IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PENMI	Res.	PEIE	Res.
	Reset value																													0		0	
0x048	RAMCFG_ M2ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	PED	Res.
	Reset value																								0							0	
0x04C	Reserved	Reserved																															
0x050	RAMCFG_ M2PEAR	BYTE[3:0]				ID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEA[15:0]															
	Reset value	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	RAMCFG_ M2ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPED	Res.
	Reset value																															0	
0x058	RAMCFG_ M2WPR1	P31WP	P30WP	P29WP	P28WP	P27WP	P26WP	P25WP	P24WP	P23WP	P22WP	P21WP	P20WP	P19WP	P18WP	P17WP	P16WP	P15WP	P14WP	P13WP	P12WP	P11WP	P10WP	P9WP	P8WP	P7WP	P6WP	P5WP	P4WP	P3WP	P2WP	P1WP	P0WP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C to 0x064	Reserved	Reserved																															
0x068	RAMCFG_ M2ERKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x06C to 0x3FC	Reserved	Reserved																															

Refer to *Section 2.3.2: Memory map and register boundary addresses* for the register boundary addresses.

7 Embedded flash memory (FLASH)

7.1 Introduction

The flash memory interface manages accesses to the flash memory, maximizing throughput to the CPU, instruction cache, and DMA. It implements the flash memory erase and program operations as well as the read and write protection mechanisms. It also implements the security and privilege access control features. It is optimized in terms of power consumption with dedicated modes when the MCU is in low-power modes.

7.2 FLASH main features

- Memory organization
  - Single bank architecture
  - Main memory: 512 Kbyte
  - Information block: 64.5 Kbyte
- Standard and burst programming modes
- Read, program, and erase operations in all voltage ranges
- 10 kcycles endurance on the whole flash memory.
- Product security activated by TrustZone® option bit (TZEN)
- Device life cycle managed by readout protection option bit (RDP)
- Two write protection areas)
- TrustZone support:
  - One secure area
  - One secure HDP (hide protection) area part of the secure area
  - One secure HDP extension area part of the secure area
- Configurable protection against unprivileged accesses with flash page granularity
- Error code correction: 8-bit ECC per 64-bit double-word allowing two bits error detection and one bit error correction
- Option byte loader
- Advanced low-power modes (low-power read mode, power-down mode)

7.3 FLASH functional description

7.3.1 Flash memory organization

The main features of the flash memory are the following:

- Single bank mode:
  - 4 Kbytes page size
  - 72-bit wide data read and write (64 effective bits plus eight ECC bits)
  - Page and mass erase

The flash memory is organized as follows:

- Main memory block organized as a single bank of 512-Kbyte containing 128 pages of 4 Kbytes.
- An information block containing:
  - 32 Kbytes for system memory. This area is immutable and reserved for use by STMicroelectronics. It contains the bootloader that is used to reprogram the flash memory through one of the user communication interfaces such as USB (DFU). The system memory is programmed by STMicroelectronics when the device is manufactured. For further details, refer to AN2606 "*STM32 microcontroller system memory boot mode*", available on *www.st.com*.
  - 32 Kbytes immutable secure area containing the root security services (RSS and RSS library) developed by STMicroelectronics
  - Up to 512 bytes OTP (one-time programmable) bytes for user data (up to 64 double-words). The OTP data cannot be erased and can be written only once. If only one bit is at 0, the entire double-word cannot be written anymore, even with the value 0x0000 0000 0000 0000.
    OTP can only be written by nonsecure access and read by secure and nonsecure access.
  - Option bytes for user configuration. Unlike user flash memory and system memory, it is not mapped to any memory address and can be accessed only through the flash register interface.

The memory is based on a main area and an information block, as detailed in *Table 43*.

Table 43. Flash module 1-Mbyte single bank organization

Flash area	Flash memory address secure(1)	Flash memory address nonsecure(1)	Size (bytes)	Name
Main memory	0x0C00 0000 - 0x0C00 0FFF	0x0800 0000 - 0x0800 0FFF	4 K	Page 0
	0x0C00 1000 - 0x0C00 1FFF	0x0800 1000 - 0x0800 1FFF	4 K	Page 1
	⋮	⋮	⋮	⋮
	0x0C07 E000 - 0x0C07 EFFF	0x0807 E000 - 0x0807 EFFF	4 K	Page 126
	0x0C07 F000 - 0x0C07 FFFF	0x0807 F000 - 0x0807 FFFF	4 K	Page 127
Information block	0x0FF8 0000 - 0x0FF8 3FFF	0x0BF8 0000 - 0x0BF8 4FFF	20 K	RSS
	0x0FF8 4000 - 0x0FF8 4FFF		4 K	RSS library
Information block	0x0FF8 5000 - 0x0FF8 CFFF	0x0BF8 5000 - 0x0BF8 CFFF	32 K	Boot loader
	0x0FF8 D000 - 0x0FF8 D1FF	0x0BF8 D000 - 0x0BF8 D1FF	512	OTP area
-	0x0FF8 D200 - 0x0FF8 D4FF	0x0BF8 D200 - 0x0BF8 D4FF	-	Reserved
Information block	0x0FF8 D500 - 0x0FF8 DFFF	0x0BF8 D500 - 0x0BF8 DFFF	2.75 K	Engineering bytes
	0x0FF8 E000 - 0x0FF8 FFFF	0x0BF8 E000 - 0xBF8 FFFF	8 K	User options

1. Gray shaded areas are reserved.

Note: The secure information block is only accessible when TrustZone is active.

7.3.2 Error code correction (ECC)

Data in flash memory are 72-bit words, as eight bits are added per double-word (64 bits). The ECC mechanism supports one-bit error detection and correction, and two-bits errors detection.

When one error is detected and corrected, the ECCC flag (ECC correction) is set in the *FLASH ECC register (FLASH_ECCR)*. If the ECCCIE bit is set, an interrupt is generated.

When two errors are detected, the ECCD flag (ECC detection) is set in the *FLASH ECC register (FLASH_ECCR)*. In this case, an NMI is generated.

When an ECC error is detected, the address of the failing double-word is saved in ADDR_ECC[19:0] in the *FLASH ECC register (FLASH_ECCR)*. ADDR_ECC[2:0] are always cleared.

When ECCC or ECCD is set, ADDR_ECC is not updated if a new ECC error occurs. FLASH_ECCR is updated only when ECC flags are cleared.

Note: For an erased flash line, only one error is detected and corrected but two errors detection is not supported.

Note: When an ECC error is reported, a new read at the failing address may not generate an ECC error if the data is still present in the current prefetch buffer or in ICACHE, even if ECCC and ECCD are cleared.

The following addresses in the flash system memory are used to store words including ECC errors to allow software runtime tests on ECC correction detection capability:

- 0x0BF8 DF00 embeds a word with 1-bit error
- 0x0BF8 DF10 embeds a word with 2-bit errors

Note: In case the second address is read, for instance by the debugger memory viewer, an NMI is generated.

7.3.3 Read access latency

To correctly read data from flash memory, the number of wait states (latency) must be correctly programmed in the *FLASH access control register (FLASH_ACR)* according to the frequency of the CPU clock (hclk1) and the internal voltage range of the device V_{CORE}. Refer to *Section 11.5.4: Dynamic voltage scaling management*.

Table 44 shows the correspondence between wait states and CPU clock frequency.

Table 44. Number of wait states according to CPU clock (hclk1) frequency (LPM = 0)

Wait states (WS) (latency)	hclk1 (MHz)		
	V_{CORE} range 1	V_{CORE} range 1.5	V_{CORE} range 2
0 WS (1 CPU cycle)	≤ 32	≤ 32	≤ 8
1 WS (2 CPU cycles)	≤ 64	≤ 64	≤ 16

The flash memory supports a low-power read mode when setting the LPM bit in the *FLASH access control register (FLASH_ACR)*. *Table 45* shows the correspondence between wait states and CPU clock frequency when LPM bit is set.

Table 45. Number of wait states according to CPU clock (hclk1) frequency (LPM = 1)

Wait states (WS) (latency)	hclk1 (MHz)	
	V_{CORE} range 1 and 1.5	V_{CORE} range 2
0 WS (1 CPU cycle)	WS ≥ hclk1 (MHz) / 10 - 1 Maximum hclk1 frequency is given by *Table 44*.	≤ 8
1 WS (2 CPU cycles)		≤ 16
2 WS (3 CPU cycles)		-
3 WS (4 CPU cycles)		-
⋮		-
6 WS (7 CPU cycles)		-

After reset, the CPU clock frequency is 16 MHz, 1 wait state (WS) is configured in the *FLASH access control register (FLASH_ACR)* and the normal read mode is selected (LPM = 0).

Before entering Stop 1, 2, and 3 mode software must set FLASH wait states to at least 1 (LATENCY) in FLASH_ACR. This to comply with the SYSCLK 16 MHz and range 2 configuration when exiting Stop 1, 2, and 3 mode.

Instruction prefetch

The Cortex-M33 fetches instructions and literal pools (constants/data) over the C-Bus and through the instruction cache if it is enabled. The prefetch block aims at increasing the efficiency of C-Bus accesses and in case the instruction cache is enabled by reducing the cache refill latency. Prefetch is efficient in case of sequential code; prefetch in the flash memory allows the next sequential instruction line to be read from the flash memory while the current instruction line is being executed by the CPU or filled in instruction cache.

Prefetch is enabled by setting the PRFTEN bit in the *FLASH access control register (FLASH_ACR)*. PRFTEN must be set only if at least one wait state is needed to access the flash memory.

Note: Prefetch tends to increase the code execution performance at the cost of extra flash memory accesses, use it carefully in low-power applications.

CPU frequency change

When changing the CPU frequency, the software sequences detailed below must be applied in order to tune the number of wait states needed to access the flash memory.

Increase the CPU frequency

1. Program the new number of wait states to the LATENCY bits in the *FLASH access control register (FLASH_ACR)*.
2. Check that the new number of wait states is taken into account to access the flash memory by reading back the *FLASH access control register (FLASH_ACR)*.
3. Modify the CPU clock source by writing the software bits in the RCC_CFGR1 register.
4. Modify the CPU clock prescaler, if needed, by writing the HPRE bits in RCC_CFGR2.
5. Check that the new CPU clock source or/and the new CPU clock prescaler value is/are taken into account by reading the clock source status (SWS bits) or/and the AHB prescaler value (HPRE bits), respectively, in the RCC_CFGR1 and RCC_CFGR2 registers.

Decrease the CPU frequency

1. Modify the CPU clock source by writing the software bits in the RCC_CFGR1 register.
2. Modify the CPU clock prescaler, if needed, by writing the HPRE bits in RCC_CFGR2.
3. Check that the new CPU clock source or/and the new CPU clock prescaler value is/are taken into account by reading the clock source status (SWS bits) or/and the AHB prescaler value (HPRE bits), respectively, in the RCC_CFGR1 and RCC_CFGR2 registers.
4. Program the new number of wait states to the LATENCY bits in the *FLASH access control register (FLASH_ACR)*.
5. Check that the new number of wait states is used to access the flash memory by reading back the *FLASH access control register (FLASH_ACR)*.

To modify the read mode apply the software sequences detailed below.

From normal read mode to low-power read mode

1. Set the LPM bit in the *FLASH access control register (FLASH_ACR)*.
2. Check that the low-power read mode is activated by reading the *FLASH access control register (FLASH_ACR)*.

From low-power read mode to normal read mode

1. Reset the LPM bit in the *FLASH access control register (FLASH_ACR)*.
2. Check that the normal read mode is activated by reading the *FLASH access control register (FLASH_ACR)*.

7.3.4 Flash power-down mode

After reset, the flash memory is in normal mode. To reduce power consumption, it can be put in power-down mode by setting the PDREQ bit in the *FLASH access control register (FLASH_ACR)*.

Request entry in power-down mode

- Check that the flash memory is not in power-down mode and no request to put it in power-down mode is pending (PD bit in *FLASH status register (FLASH_NSSR)* and PDREQ bit in *FLASH access control register (FLASH_ACR)* must be reset).
- Unlock PDREQ write access in PDKEYR with the correct keys. Refer to *FLASH power-down key register (FLASH_PDKEYR)*
- Set the PDREQ bit in the *FLASH access control register (FLASH_ACR)*.
- Check that the PD bit in the *FLASH power-down key register (FLASH_PDKEYR)* is set. The PDREQ bit in the *FLASH access control register (FLASH_ACR)* is automatically reset and write access is locked.

Note: If the memory is being accessed, the power-down request is delayed until the access is completed.

Requesting power-down entry for the flash memory already in power-down mode has no effect. The PDREQ bit in the *FLASH access control register (FLASH_ACR)* is automatically reset and the PDKEYR is locked.

Return to normal mode

Any access to flash memory in power-down mode automatically wakes up the memory, at least 5 µs are needed to wake it up.

The flash memory wake up is done in one of the following cases:

- Upon a valid read access to flash memory
- Upon a valid write access to flash memory
- Upon a valid mass erase
- Upon an option byte modification
- Upon an option byte loading
- Upon system reset

Note: The software can reduce the flash wake-up time by enabling HSI16 before waking up the flash.

7.3.5 Flash memory program and erase operations

The embedded flash memory can be programmed using in-circuit programming (ICP) or in-application programming (IAP).

The ICP method is used to update the entire contents of the flash memory, using the JTAG, SWD protocol, or the bootloader to load the user application into the microcontroller. ICP offers quick and efficient design iterations and eliminates unnecessary package handling or socketing of devices.

In contrast to the ICP method, the IAP can use any communication interface supported by the microcontroller (such as I/Os, UART, I2C, SPI, USB, or 2.4 GHz RADIO) to download programming data into the memory. IAP allows the user to reprogram the flash memory while the application is running. Nevertheless, part of the application must have been previously programmed in the flash memory using ICP.

An ongoing flash memory operation halts the CPU when accessing flash. The CPU operation from flash proceeds correctly once the program/erase operation is completed.

Program and erase operations can be suspended to guarantee flash read and execution access for real-time critical software.

The MCU supports Trustzone that defines secure and nonsecure areas in the flash memory. Program and erase operations can be performed in secure mode through the secure registers or in nonsecure mode through the nonsecure registers. For more information, refer to *Section 7.5*.

Unlock the secure/nonsecure flash control registers

After reset, write is not allowed in the flash control registers (*FLASH secure control register (FLASH_SECCR1)* and *FLASH control register (FLASH_NSCR1)*) in order to protect the flash memory against possible unwanted operations due, for example, to electric disturbances.

The following sequence is used to unlock these registers:

1. Write KEY1 = 0x45670123 in the *FLASH secure key register (FLASH_SECKEYR)* or *FLASH key register (FLASH_NSKEYR)*.
2. Write KEY2 = 0xCDEF89AB in the *FLASH secure key register (FLASH_SECKEYR)* or *FLASH key register (FLASH_NSKEYR)*.

Any wrong sequence locks up the *FLASH secure control register (FLASH_SECCR1)* or *FLASH control register (FLASH_NSCR1)* register until the next system reset. In the case of a wrong key sequence, a bus error is detected and a hard fault interrupt is generated.

The FLASH_NSCR1 (resp. FLASH_SECCR1) register can be locked again by software by setting the LOCK bit in the FLASH_NSCR1 (resp. FLASH_SECCR1) register.

*Note: The FLASH*_NSCR1 and FLASH_SECCR1 *registers cannot be written when the BSY bits are set. Any attempt to write them with the BSY bits set, causes the write to be ignored and generated a bus error and hard fault interrupt.*

Wait for data-to-write flags (WDW)

The WDW flags in the *FLASH status register (FLASH_NSSR)* and *FLASH secure status register (FLASH_SECSR)* are both set when a secure or nonsecure write access has been done in the write buffer. They are cleared when the BSY flags are set (meaning that the write buffer is freed and the programming operation actually starts in the flash memory) or in case of error.

It is the software responsibility to ensure that the two words in the same double-word are all written.

Secure and nonsecure busy flags

The BSY flags in the *FLASH status register (FLASH_NSSR)* and *FLASH secure status register (FLASH_SECSR)* are both set when a secure or nonsecure flash operation is started:

- Erase operation: setting the STRT bit in the *FLASH control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)*
- Write operation: setting the PG bit in the *FLASH control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)* and writing a double-word in the flash memory
- Option bytes programming: setting the OPTSTRT in the *FLASH control register (FLASH_NSCR1)*

7.3.6 Flash memory erase sequences

The flash memory erase operation can be performed at page level or on the whole memory (mass erase). Mass erase does not affect the information block (system flash, OTP, and option bytes). The erase operation is either secure or nonsecure.

Page erase

A page erase is possible only in two cases:

- When a page is nonsecure, by a nonsecure page erase request through *FLASH control register (FLASH_NSCR1)*
- When a page is secure, by a secure page erase request through *FLASH secure control register (FLASH_SECCR1)*

To erase a page, follow the procedure below:

1. Check that no flash memory operation is ongoing by checking the BSY bit in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)* and that the write buffer is empty by checking the WDW bit in FLASH_SR or FALSH_SSR.
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Set the PER bit and select the page to erase (PNB) in the *FLASH control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)*.
4. Set the STRT bit in the *FLASH control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)*.
5. Wait for the BSY bit to be cleared in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.
6. Clear PER bit can be cleared if no more page erase is requested.

Mass erase

A mass erase is possible only in two cases:

1. When full flash is nonsecure, by a nonsecure mass erase request through *FLASH control register (FLASH_NSCR1)*
2. When full flash is secure, by a secure mass erase request through *FLASH secure control register (FLASH_SECCR1)*

To perform a mass erase, follow the procedure below:

1. Check that no flash memory operation is ongoing by checking the BSY bit in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register*

(FLASH_SECSR) and that the write buffer is empty by checking the WDW bit in FLASH_SR or FALSH_SSR.

2. Check and clear all nonsecure error programming flags due to a previous programming. If not, the PGSERR bit is set.
3. Set the MER bit in the *FLASH control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)*.
4. Set the STRT bit in the *FLASH control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)*.
5. Wait for the BSY bit to be cleared in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.
6. The MER bit can be cleared if no more mass erase is requested.

Note: Page and mass erase are only started when erase operations have not been suspended by ES in FLASH control 2 register (FLASH_NSCR2) or FLASH secure control 2 register (FLASH_SECCR2).

Note: The internal oscillator HSI16 (16 MHz) is enabled automatically when the STRT bit is set, and disabled automatically when the STRT bit is cleared, except if the HSI16 is previously enabled with HSION in RCC_CR register.

To erase a page or to perform a mass erase, the software must have sufficient privilege (see Table 65 and Table 66).

7.3.7 Flash memory programming sequences

The flash memory is programmed 72 bits at a time (64-bit data + 8-bit ECC).

Programming in a previously programmed address is not allowed, except if the data to write is full zero, any attempt sets the PROGERR flag in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.

It is possible only to program double-word (2 x 32-bit data).

- Any attempt to write a byte or a half-word sets the SIZERR flag in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.
- Any attempt to write a double-word that is not aligned with a double-word address sets the PGAERR flag in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.

Flash memory programming

Flash memory programming is possible only in two cases:

- When the program location is nonsecure, by a nonsecure program request through *FLASH control register (FLASH_NSCR1)* and nonsecure double-word write sequence
- When the program location is secure, by a secure program request through *FLASH secure control register (FLASH_SECCR1)* and secure double-word write sequence.

The programming sequence is as follows:

1. Check that no flash main memory operation is ongoing by checking the BSY bit in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.
2. Check that the write buffer is empty by checking the WDW bit in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.
3. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
4. Set the PG bit in the *FLASH control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)*.
5. Perform the data write operation at the desired flash memory address, or in the OTP area. Only a double-word can be programmed:
   - Write a first word in an address aligned on a double-word address. The WDW bits in the *FLASH status register (FLASH_NSSR)* and *FLASH secure status register (FLASH_SECSR)* are set to indicate that more data can be written in the write buffer.
   - Write the second word in the same double-word.
6. The BSY bit gets set. WDW is reset automatically.
7. Wait until both the WDW bit and BSY bit are cleared in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*. The software must first check that the WDW is cleared before checking that the BSY is cleared.
8. If the EOP flag is set in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)* (meaning that the programming operation has succeeded and the EOPIE bit is set), it must be cleared by software.
9. Clear the PG bit in the *FLASH control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)* if there is no more programming request.

Note: Flash program operations start only when program operations have not been suspended by PS in FLASH control 2 register (FLASH_NSCR2) or FLASH secure control 2 register (FLASH_SECCR2).

Note: When the flash memory interface receives a good program sequence (a double-word), programming is automatically launched and the BSY bits are set. The internal oscillator HSI16 (16 MHz) is enabled automatically when PG bit is set, and disabled automatically when PG bit is cleared, except if the HSI16 is previously enabled with HSION in RCC_CR register.

Option bytes modifications or erase requests are not allowed when the WDW bit is set.

Programming is possible only if the privileged and security attributes are respected. Refer to Section 7.7.

If the user needs to program only one word, the double-word must be completed with the erase value 0xFFFF FFFF to launch automatically the programming.

The OTP data cannot be erased and is writable only once. If one bit is at 0, the entire double-word cannot be written anymore, neither with the value 0x0000 0000 0000 0000.

ECC is calculated from the double-word to program.

Burst programming (16 double-words)

This programming is possible only in two cases:

- When the program location is nonsecure, by a nonsecure program request through *FLASH control register (FLASH_NSCR1)* and nonsecure double-word write sequence.
- When the program location is secure, by a secure program request through *FLASH secure control register (FLASH_SECCR1)* and secure double-word write sequence.

The burst programming sequence is as follows:

1. Check that no operation is ongoing in the flash main memory by checking the BSY bit in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.
2. Check that the write buffer is empty by checking the WDW bit in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.
3. Check and clear all error programming flags due to previous programming(s). If none, PGSERR is set.
4. Set the BWR and PG bits in the *FLASH control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)*.
5. Perform the data write operation at the desired flash memory address, or in the OTP area. Only 16 double-words can be programmed:
   - Write a first 32-bit word in an address aligned on a 16 x double-word address (multiple of 0x80). The WDW bits in the *FLASH status register (FLASH_NSSR)* and *FLASH secure status register (FLASH_SECSR)* are set to indicate that more data can be written in the write buffer.
   - Write the 31 other 32-bit words consecutively.
6. The BSY bit gets set. WDW is reset automatically.
7. Wait until both the WDW bit and BSY bit are cleared in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*. The software must first check that the WDW bit is cleared before checking that the BSY bit is cleared.
8. If the EOP flag is set in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)* (meaning that the programming operation has been successful and the EOPIE bit is set), it has to be cleared by software.
9. Clear the BWR and PG bits in the *FLASH control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)* if there is no more programming request.

Note: The program operation can only start when the program operations have not been suspended by PS in FLASH control 2 register (FLASH_NSCR2) or FLASH secure control 2 register (FLASH_SECCR2).

Note: When the flash memory interface receives a good sequence, programming is automatically launched and the BSY bits are set. The internal oscillator HSI16 (16 MHz) is enabled automatically when PG bit is set, and disabled automatically when PG bit is cleared, except if the HSI16 is previously enabled with HSION in RCC_CR register.

No option bytes modification nor erase request is allowed when the WDW bit is set.

Programming is possible only if the privileged and security attributes are respected. Refer to Section 7.7.

7.3.8 Flash memory programming erases suspend

To prevent real-time critical program execution from being interrupted by flash main memory program or erase operations, these operations can be suspended during the real time program execution phase. For this, a program and a erase suspend feature is available which prevents any new program or erase operation to be started. These operations are suspended (delayed) until after the suspend feature has been disabled. During the suspend period and after any ongoing program or erase operation has been completed, program execution and memory read operation from flash are still available.

Due to the difference in timing constrains for program and erase operation, these can be suspended separately by the program suspend register PS and ES bits in *FLASH control 2 register (FLASH_NSCR2)* and *FLASH secure control 2 register (FLASH_SECCR2)*, where PS suspends any new program operation and ES suspends any new erase operation.

Furthermore, program and erase suspend is available as secure and nonsecure request, resulting in possibility of separate program and erase suspend control from the secure and nonsecure sides.

When suspending flash programming by setting a PS bit the following happens:

- Any ongoing secure or nonsecure program operation, started before the PS has been set, completes. The maximum latency for a flash main memory program suspend to guarantee uninterrupted execution from flash is the time to program the flash (see datasheet for times).
- A new requested secure and nonsecure program operation is suspended until all PS bits are cleared. The suspended program operation has the BSY bit set from the moment the last double-word has been written in a flash program sequence or the last of the 32 words in a flash burst program sequence.

Note: Program suspend is applied to both flash main memory and OTP programming.

Warning: When programming is suspended and a new program operation is requested after any good program sequence, any additional flash write or write to FLASH_NSCR1 and FLASH_SECCR1 halts the bus master (CPU or DMA) that writes.

When suspending flash erase by setting an ES bit the following happens:

- Any ongoing secure or nonsecure erase operation, started before the ES has been set, completes. The maximum latency for a flash main memory erase suspend to guarantee uninterrupted execution from flash is the time to erase the memory (see datasheet for times).
- A new requested secure and nonsecure erase operation is suspended until all ES bits are cleared. The suspended erase operation has the BSY bit set from the moment the erase STRT bit is set. Both page erase and mass erase are suspended.

Note: Erase suspend has no effect when the flash memory is erased by RDP regression. Flash erase requested via RDP regression is performed even when ES bits are set.

Table 46. Program and erase suspend control

PS		ES		Description
Secure	Nonsecure	Secure	Nonsecure	
0	0	0		Program and erase operation granted
Not 00				Any new start of program operation suspended Erase operations granted
0	0	Not 00		Program operations granted Any new start of erase operation suspended
Not 00				Any new start of program or erase operation suspended

7.3.9 Flash memory operation abort

Flash operations mass erase, page erase, and burst write can be aborted to allow time critical flash use to take priority.

- An aborted erase operation must be reissued for the remaining erase time.
- An aborted burst write must be reinitiated after the last correctly written double-word by using single write programming. The last correctly written double-word is reported in ET_OP.

An ongoing flash operation is aborted with register bit OPA. This bit can be set by software running in SRAM or by a hardware interrupt enabled in SYSCFG. Up on an aborted flash operation the OPBRK flag is set. For aborted secure operations the secure OPBRK flag is set, and for aborted nonsecure flash operations the nonsecure OPBRK flag is set.

Secure flash operations can only be aborted by a secure event using the secure register bit OPA.

Nonsecure flash operations can be aborted by both secure and nonsecure events using the secure or nonsecure register bit OPA.

The abortion of a flash operation will provide its status in register FLASH_OPSR. The elapsed erase time or burst correctly programmed double-words can be read back from the register field ET_OP. FLASH_OPSR shall be read before clearing the OPBRK flag.

See the datasheet for flash operation abort times.

7.3.10 Flash memory endurance

Each memory page can be written and erased 10000 times. In addition, up to 256 Kbytes (32 pages) feature an increased endurance of 100 kcycles, which can be used for data storage that usually needs more intensive cycling capability compared to code storage. Any page can be chosen to be cycled more than 10000 times (up to 100000 times). It is the application responsibility to limit the size of the flash area cycled more than 10000 times to these 32 pages.

7.3.11 Flash memory errors flags

Flash programming errors

Several kinds of error can be detected during secure and nonsecure operations. In case of error, the flash memory operation (programming or erasing) is aborted.

The secure errors flags are only set during a secure operation and nonsecure flags are only set during a nonsecure operation.

- **PROGERR**: secure/nonsecure programming error

  It is set when the word to program is pointing to an address:
  - Not previously erased
  - Already fully programmed to 0
  - Already programmed and the new value to program is not full 0
  - For OTP programming, when the address is already programmed
- **SIZERR**: secure/nonsecure size programming error

  Only 32-bit data can be written. The SIZERR flag is set if a byte or a half-word is written.
- **PGAERR**: secure/nonsecure alignment programming error

  It is set when the first word to be programmed is not aligned with a double-word address, or the second, third or forth word does not belong to the same double-word address.

  For burst programming, it is set when the first word to be programmed is not aligned on a 16 x double-word address or if the following word writes are not done at consecutive 32-bit addresses.
- **PGSERR**: programming sequence error

  PGSERR is set if one of the following conditions occurs during an erase or program operation:
  - A data is written when PG is cleared.
  - A program operation is requested during erase: PG is set while MER or PER is set.
  - In the erase sequence, PG is set while STRT is already set.
  - In the erase sequence, if STRT is set while MER and PER are cleared.
  - If page and mass erase are requested at the same time, STRT and PER are set while MER is set.
  - If an operation is started while the write buffer is waiting for the next data, STRT or OPTSTRT is set while WDW is already set.
  - If STRT and OPTSTRT are set at the same time.
  - A nonsecure PGSERR is set if the nonsecure STRT bit is set by a secure access.
  - A secure PGSERR is set if PROGERR, SIZERR, PGAERR, WRPERR, or PGSERR is already set due to a previous programming error.
  - A nonsecure PGSERR is set if PROGERR, SIZERR, PGAERR, WRPERR, PGSERR or OPTWERR is already set due to a previous programming error.
- **WRPERR**: write protection error
  - Refer to *Table 62* to *Table 66* for the conditions of WRPERR flag setting. In addition, the WRPERR flag is set when a user option programming starts with OPTSTRT in RDP level 2.

- **OPTWERR**: option bytes write error

  OPTWERR is set if when user option bytes are modified with an invalid configuration. It is set when attempting:

  – To program an invalid secure watermark-based area. Refer to *Table 50*
  – To set or clear the TZEN option bit when RDP is not at the correct level (refer to *Rules for modifying specific option bytes*)
  – To clear the BOOT_LOCK option bit when RDP is not at the correct level (refer to *Rules for modifying specific option bytes*)
  – To modify the SECBOOTADD0 option bit while BOOT_LOCK is set
  – To modify SECWMR2 while HDP_ACCDIS is disabled or HDPEXT_ACCDIS is disabled.
  – To modify the option bytes, when RDP is level 2
  – To regress from RDP level 0.5 to RDP level 0
  – To modify OEM1KEYRx while the RDP level is 0.5 or 1 and OEM1LOCK bit is set
  – To modify OEM2KEYRx while the RDP level is 1 and the OEM2LOCK bit is set
  – To regress from RDP level 1 to RDP level 0 while the OEM1LOCK bit is set and a wrong OEM1 key is shifted through JTAG
  – To regress from RDP level 1 to RDP level 0.5 while the OEM2LOCK bit is set and a wrong OEM2 key is shifted through JTAG
  – To program OEM1KEY with less than five zero.
  – To program OEM2KEY with less than five zero.
  – To modify WRPxyR while its UNLOCK bit is cleared
  – To set the UNLOCK bit in the WRPxyR when the RDP is not at the correct level (refer to *Rules for modifying specific option bytes*)

If an error occurs during a secure or nonsecure program or erase operation, one of the following programming error flags is set:

- nonsecure programming error flags: PROGERR, SIZERR, PGAERR, PGSERR, OPTWRERR, or WRPERR is set in the *FLASH status register (FLASH_NSSR)*.

  If the nonsecure error interrupt enable bit ERRIE is set in the *FLASH control register (FLASH_NSCR1)*, an interrupt is generated and the operation error flag OPERR is set in the FLASH_NSSR register.
- Secure programming error flags: PROGERR, SIZERR, PGAERR, PGSERR, or WRPERR is set in the *FLASH secure status register (FLASH_SECSR)*.

  If the secure error interrupt enable bit ERRIE is set in the *FLASH secure control register (FLASH_SECCR1)*, an interrupt is generated and the operation error flag OPERR is set in the FLASH_SECSR register.

Note: If several successive errors are detected (for example, in case of DMA transfer to the flash memory), the error flags cannot be cleared until the end of the successive write requests. Any programming error flushes the write buffer.

7.3.12 Power-down during programming or erase operations

The contents of the flash memory currently being accessed are not guaranteed if a power-down occurs during a flash memory program or erase operation.

7.3.13 Interruption during programming or erase operations

The contents of the flash memory currently being accessed during a flash memory program or erase operation are not guaranteed if an interruption (system reset or abort) occurs. On an interruption during a flash memory program or erase operation the status of the flash memory can be recovered from the CODE_OP in *FLASH operation status register (FLASH_OPSR)*, and the associated accessed address can be recovered from ADDR_OP.

It is the software responsibility to check the status of the flash memory and to take corrective actions. This is advised to be done after each system reset and at the end of any flash operation, before performing any other programming or erase operation.

After a flash operation interruption due to reset, CODE_OP and ADDR_OP are reinitialized when starting a new program or erase operation. When a program or erase operation completes successfully CODE_OP and ADDR_OP are 0

After a flash operation interruption due to abort, CODE_OP and ADDR_OP are reinitialized when both OPBRK bits are zero.

Table 47 describes the corrective action to be taken according to the status provided in the CODE_OP field of the *FLASH operation status register (FLASH_OPSR)*.

Table 47. Flash operation interrupted by a system reset

CODE_OP	Operation interrupted	Address	System flash	Corrective action
0x0	No operation	Reserved		None
0x1	Single write	ADDR_OP	SYSF_OP	Page erase and single write at same location
0x2	Burst write	ADDR_OP	SYSF_OP	Due to system reset: Page erase and burst write at same location Due to operation abort: re-initiate single write programming after the last correctly written double-word.
0x3	Page erase	ADDR_OP	Reserved	Erase same page
0x4	Reserved			
0x5	Mass erase	Reserved		Mass erase
0x6	Option change	Reserved		Option change
0x7	Reserved			

Note: For single and burst write interruption due to a system reset, it is mandatory to perform a page erase because the flash memory locations with interrupted program operation, may no longer be writable and on read may generate an ECC error. Consequently, the remaining page content must be saved before performing a page erase and restored afterwards.

For OTP locations, it is not possible to perform a page erase. An OTP double-word with interrupted program operation is lost.

For burst write, ADDR_OP gives the first address of burst. After reset, the user must restart the same burst operation. After abort, the user may program aborted double-words with a single write starting after the last correctly programed double-word reported in ET_OP.

For page erase, ADDR_OP gives the first address of the erased page.

7.4 FLASH option bytes

7.4.1 Option bytes description

The option bytes are configured by the end user depending on the application requirements. As a configuration example, the watchdog may be selected in hardware or software mode (refer to *Section 7.4.2*). The user option bytes are accessible through the flash memory registers interface.

Table 48 describes the organization of the user option bytes available in the flash memory interface registers.

Table 48. User option byte organization mapping

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	Register map
TZEN	Res.	Res.	Res.	NBOOT0	NSWBOOT0	SRAM2_RST	SRAM2_PE	Res.	Res.	Res.	Res.	Res.	IWDG_STDBY	IWDG_STOP	IDWG_SW	SRAM1_RST	Res.	NRST_STDBY	NRST_STOP	Res.	BOR_LEV[2:0]			RDP[7:0]								*Section 7.9.13*
NSBOOTADD0[24:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.		*Section 7.9.16*
NSBOOTADD1[24:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.		*Section 7.9.17*
SECBOOTADD0[24:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT_LOCK	*Section 7.9.18*
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM_PEND[6:0]															SECWM_PSTRT[6:0]								*Section 7.9.19*
HDPEN[7:0]								Res.	HDP_PEND[6:0]															Res.	Res.	Res.	Res.	Res.	Res.	Res.		*Section 7.9.20*
UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPA_PEND[6:0]															WRPA_PSTRT[6:0]								*Section 7.9.20*
UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPB_PEND[6:0]															WRPB_PSTRT[6:0]								*Section 7.9.21*
OEM1KEY[31:0]																																*Section 7.9.27*
OEM1KEY[63:32]																																*Section 7.9.28*
OEM1KEY[95:64]																																*Section 7.9.29*
OEM1KEY[127:96]																																*Section 7.9.30*
OEM2KEY[31:0]																																*Section 7.9.31*
OEM2KEY[63:32]																																*Section 7.9.32*
OEM2KEY[95:64]																																*Section 7.9.33*
OEM2KEY[127:96]																																*Section 7.9.34*

7.4.2 Option bytes programming

Unlocking option control register

After reset, the options related operation bits OPTSTRT and OBL_LAUNCH in the FLASH_NSCR1 register are write-protected. To run any operation on the option bytes page,

the option lock bit OPTLOCK in the *FLASH control register (FLASH_NSCR1)* must be cleared.

The following sequence is used to unlock this register:

1. Unlock the FLASH_NSCR1 register with the LOCK clearing sequence (refer to *Unlock the secure/nonsecure flash control registers*).
2. Write OPTKEY1 = 0x08192A3B in the FLASH_OPTKEYR register.
3. Write OPTKEY2 = 0x4C5D6E7F in the FLASH_OPTKEYR register.

The user options can be protected against unwanted erase/program operations by setting the OPTLOCK bit by software.

Note: If the LOCK bit in FLASH_NSCR1 *is set by software, OPTLOCK is automatically set too.*

Option bytes modification sequence

To modify the user options value, follow the procedure below:

1. Check that no flash memory operation is in progress by checking the BSY bit in the FLASH_NSSR register.
2. Clear OPTLOCK option lock bit with the clearing sequence described above.
3. Write the desired options value in the options registers.
4. Set the options start bit OPTSTRT in the FLASH_NSCR1 register.
5. Wait for the BSY bit to be cleared.
6. Disable the LSE oscillator by setting LSEON to 0 and waiting for LSERDY to be 0.
7. Set the OBL_LAUNCH option bit in the FLASH_NSCR1 register to start option bytes loading.

Note: If the OPTWERR or PGSERR error bit is set, the old option byte values are kept.

*Note: The FLASH_*NSCR1 *register cannot be written when the OBL_LAUNCH bit is set. Any attempt to write it with the OBL_LAUNCH bit set, causes the write to be ignored and generated a bus error and hard fault interrupt.*

Option byte loading

After the BSY bit is cleared, all new options are updated into the flash memory but they are not applied to the system. They affect the system when they are loaded. Option bytes loading (OBL) is performed in two cases:

- When OBL_LAUNCH bit is set in the *FLASH control register (FLASH_NSCR1)*
- After a power reset (BOR0 reset) or exit from Standby modes

After a system reset, internal option registers are copied into option registers and factory-programmed values are loaded. The internal option registers are also used to modify the option bytes. If these registers are not modified by the user, they reflect the options state of the system.

Note: The factory-programmed value for LSETRIM is only loaded when SBF is cleared. They are not loaded on exit from Standby modes. See Section 12.4.4: LSE clock and Section 11.7.10: PWR Standby mode for more information on these features.

Rules for modifying specific option bytes

Some of the option byte field must respect specific rules before being updated with new values. These option bytes, as well as the associated constraints, are described below:

- TZEN option bit
  - TZEN can only be set on RDP level 0.
  - Deactivation of TZEN is possible only when RDP is changing from level 1 to level 0.
- BOOT_LOCK option bit
  - BOOT_LOCK has only effect when TZEN is set.
  - BOOT_LOCK can be set without any constraint.
  - Deactivation of BOOT_LOCK is possible only when RDP is changing from level 1 to level 0 or when RDP is in level 0.
- SECBOOTADD0 option bytes
  - It cannot be modified when the BOOT_LOCK option bit is set.
- SECWMRy option bits
  - It is not possible to modify secure (SECWM_PSTRT and SECWM_PEND option bits) and HDP (HDP_PEND and HDPEN option bits) area when the HDP access (HDP_ACCDIS) is disabled or the HDP extension access (HDPEXT_ACCDIS) is disabled.
- RDP option bits
  - Depending on the OEMx key RDP lock state, the following regressions are blocked: from RDP level 2 to level 1, from RDP level 1 to level 0.5 or from level 1 to level 0. Refer to *Device life cycle managed by readout protection (RDP) transitions*.
- WRPxR option bits
  - These bits cannot be modified when their UNLOCK bit is cleared.
- UNLOCK option bits
  - These bits can be set only when regressing from RDP level 1 to level 0.

If the user options modification tries to set or modify one of the listed option bytes without following their associated rules, the option bytes modification is discarded and the OPTWERR error flag is set.

7.5 FLASH TrustZone security and privilege protections

7.5.1 Trustzone security protection

The global TrustZone system security is activated by setting the TZEN option bit in the FLASH_OPTR register.

When TrustZone is active (TZEN = 1), the following additional security features are available:

- Secure watermark-based user options bytes defining a secure, HDP area.
- Secure HDP extension area.
- Secure or nonsecure block-based areas can be configured on-the-fly after reset. This is a volatile secure area.
- An additional RDP protection: RDP level 0.5
- Erase or program operation can be performed in secure or nonsecure mode with associated configuration bit.

When the TrustZone is disabled (TZEN = 0), the above features are deactivated and all secure registers are RAZ/WI.

Activate TrustZone security

When the TrustZone is activated (TZEN is modified from 0 to 1), the secure watermark-based user options bytes are set to the default secure state: all flash memory is secure, no HDP area as shown in *Table 49*.

Table 49. Default secure option bytes after TZEN activation

Secure watermark option bytes values after OBL when TZEN is activated (from 0 to 1)	Security attribute
SECWM_PSTRT = 0 and SECWM_PEND = 0x7F	All flash memory secure
HDPEN = 0xB4 and HDP_PEND = 0	No secure HDP area

Illegal access generation

A nonsecure access to a secure flash memory area is RAZ/WI and generates an illegal access event. An illegal access interrupt is generated if the FLASHIE illegal access interrupt is enabled in the GTZC1_TZIC_IERx register.

A nonsecure access to a secure flash register is RAZ/WI and generates an illegal access event. An illegal access interrupt is generated if the FLASH_REGIE illegal access interrupt is enabled in the GTZC1_TZIC_IERx register.

Deactivate TrustZone security

Deactivation of TZEN (from 1 to 0) is possible only when the RDP is changing from level 1 to level 0.

When the TrustZone is deactivated (TZEN is modified from 1 to 0) after option bytes loading, the following security features are deactivated:

- Watermark-based secure area (refer to *Section 7.5.2*)
- Block-based secure area (refer to *Section 7.5.5*)
- RDP level 0.5 (refer to *Section 7.6.2*)
- Secure interrupts (refer to *Section 7.8*)
- All secure registers are RAZ/WI

7.5.2 Watermark-based secure flash memory area protection

When TrustZone security is active (TZEN = 1), a part of the flash memory can be protected against nonsecure read and write accesses. One nonvolatile secure areas can be defined by option bytes and can be read or written by a secure access only: one area can be selected with a page granularity.

The secure areas are defined by a start-page offset and end-page offset using the SECWM_PSTRT and SECWM_PEND option bytes. These offsets are defined in the secure watermark address registers *FLASH secure watermark register 1 (FLASH_SECWMR1)*.

The SECWM_PSTRT and SECWM_PEND option bytes can only be modified by secure firmware when the HDP_ACCDIS and HDPEXT_ACCDIS values are reset (0xA3). If the HDP_ACCDIS or HDPEXT_ACCDIS is different than the reset value, the SECWM_PSTRT and SECWM_PEND cannot be modified until the next system reset.

Table 50. Secure watermark-based area

Secure watermark option bytes values (x = 1, 2)	Secure watermark protection area
SECWM_PSTRT > SECWM_PEND	No secure area
SECWM_PSTRT = SECWM_PEND	One page defined by SECWM_PSTRT is secure watermark-based protected
SECWM_PSTRT < SECWM_PEND	The area between SECWM_PSTRT and SECWM_PEND is secure watermark-based protected.

Caution: Switching a flash memory area from secure to no-secure does not erase its content. The user secure software must perform the needed operation to erase the secure area before switching an area to nonsecure attribute whenever it is needed. It is also recommended to flush the instruction cache.

7.5.3 Secure hide protection (HDP)

The secure HDP area is part of the flash memory watermark-based secure area. Access to the hide protection area can be denied by setting the HDP_ACCDIS or HDPEXT_ACCDIS to a value different than 0xA3 in the *FLASH secure HDP control register (FLASH_SECHDPCR)*.

When the HDP_ACCDIS or HDPEXT_ACCDIS is disabled (value different than 0xA3), instruction fetch, data read, write, and erase operations on these hide protection areas are denied. For example, software code in the secure flash hide protected areas can be executed only once and deny any further access to this area until the next system reset or

standby reset. The HDP_ACCDIS or HDPEXT_ACCDIS values can be cleared only by a system reset and standby reset.

When a secure HDP area is enabled (HDPEN different from 0xB4 and SECWM_PSTRT ≤ HDP_PEND ≤ SECWM_PEND), it is forbidden to fetch outside the HDP area, until he HDP access is disabled in HDP_ACCDIS.

Note: It is the software responsibility to take any appropriate action to protect the HDP code before disabling the HDPEN such as erasing the HDP area and flushing the instruction cache.

One flash secure hide protection (HDP) area can be defined with flash page granularity.

The secure HDP area is enabled by the HDPEN. When the HDPEN is disabled, there is no HDP area. The HDPEN value can be changed on the fly by the secure firmware when the HDP_ACCDIS is enabled. If the HDP_ACCDIS is disabled the HDPEN and secure watermark configuration cannot be modified until the next system reset.

The secure HDP area size is defined by the end-page offset using the HDP_PEND option bytes while the start-page offset is already defined by SECWM_PSTRT option bytes. These offsets are defined in the secure watermark address registers: *FLASH secure watermark register 1 (FLASH_SECWMR1)*, and *FLASH secure watermark register 2 (FLASH_SECWMR2)*.

For example, to protect by HDP from the address 0x0C00 4000 (included) to the address 0x0C00 5FFF (included):

- The option bytes registers must be programmed with:
  - SECWM_PSTRT = 0x2
  - HDP_PEND = 0x3

If an invalid secure HDP area is defined (as described in *Table 51*), the OPTWERR flag error is set and option bytes modification is discarded.

Table 51. Secure hide protection

HDP and secure watermark option bytes values		Hide protection area
HDPEN = 0xB4	-	No secure HDP area
HDPEN different 0xB4	SECWM_PSTRT ≤ HDP_PEND ≤ SECWM_PEND	The area between SECWM_PSTRT and HDP_PEND is secure HDP protected.
	Others	Invalid secure area. Hide protection area is defined outside the secure area.

7.5.4 Secure hide protection extension (HDP extension)

The secure HDP area can be extended using HDP_PEXT values in FLASH_SECHDPEXTR indicates the number of pages added for the HDP extension area.

The HDP extension is composed only by the secure HDP_PEXT pages which belong to the flash memory watermark-based secure area. Any page outside the watermark-based secure area is not part of the HDP extension area. That is to say the end page offset of the HDP extension area cannot be greater than SECWM_PEND.

The start-page offset of the HDP extension area depends on the presence of an HDP area. If there is an HDP area, the HDP extension area starts from HDP_PEND + 1. Otherwise it starts from SECWM_PSTRT.

Note: If there is no watermark-based secure area, there is no HDP extension area.

An HDP extension area can be defined even if there is on valid HDP area.

Access to the HDP extension area can be denied thanks to HDPEXT_ACCDIS in FLASH_SECHDPCR.

- When HDPEXT_ACCDIS value has the reset value (0xA3), access to the HDP extension is granted. There is no write access restriction to HDP_PEXT and HDPEXT_ACCDIS.
- When HDPEXT_ACCDIS value is set to 0x5C, any operation (instruction fetch, data read, write, and erase) to the HDP extension or HDP is denied until the next system reset or standby reset. Moreover HDP extension area size can only be incremented. There are some write access restrictions on HDP_PEXT and HDPEXT_ACCDIS.
  - HDP_PEXT can only be incremented
  - HDPEXT_ACCDIS cannot be set to 0xA3
- When HDPEXT_ACCDIS value is set to a value different than 0xA3 and 0x5C, any operation (instruction fetch, data read, write, and erase) to the HDP extension area and HDP area is denied until the next system reset or standby reset. Moreover HDP extension area size cannot be modified until the next system reset or standby reset. There are some write access restrictions on HDP_PEXT and HDPEXT_ACCDIS:
  - Write access to HDP_PEXT bits are ignored
  - Write access to HDPEXT_ACCDIS is ignored

Table 52. Write access to HDP_PEXT and HDPEXT_ACCDIS

Current HDPEXT_ACCDIS	Write access to		Operation (fetch, read, write, erase) in HDP extension area	Operation (fetch, read, write, erase) in HDP area
	HDPEXT_ACCDIS	HDP_PEXT		
0xA3 (reset value)	allowed	allowed	allowed	depend on HDP_ACCDIS
0x5C	allowed if new value different than 0xA3. 0xA3 ignored	allowed if new value is greater than current value, otherwise ignored	denied	denied
other	ignored	ignored		

When the HDP extension is enabled (HDPEXT_ACCDIS different from 0xA3), the HDPEN and secure watermark configuration cannot be modified until the next system reset or standby reset.

One flash secure hide protection extension area can be defined with flash page granularity.

When an HDP extension area is defined and its access is still enabled (HDPEXT_ACCDIS = 0xA3) and the HDP area access is disabled, it is forbidden to fetch outside the HDP extension area until its access is disabled. The HDP and HDP extension area protections are summarized in *Table 61: Flash memory secure access versus HDP and HDP extension (TZEN = 1)*.

The table below summarizes the possible HDP extension area configuration.

Table 53. Secure hide protection extension

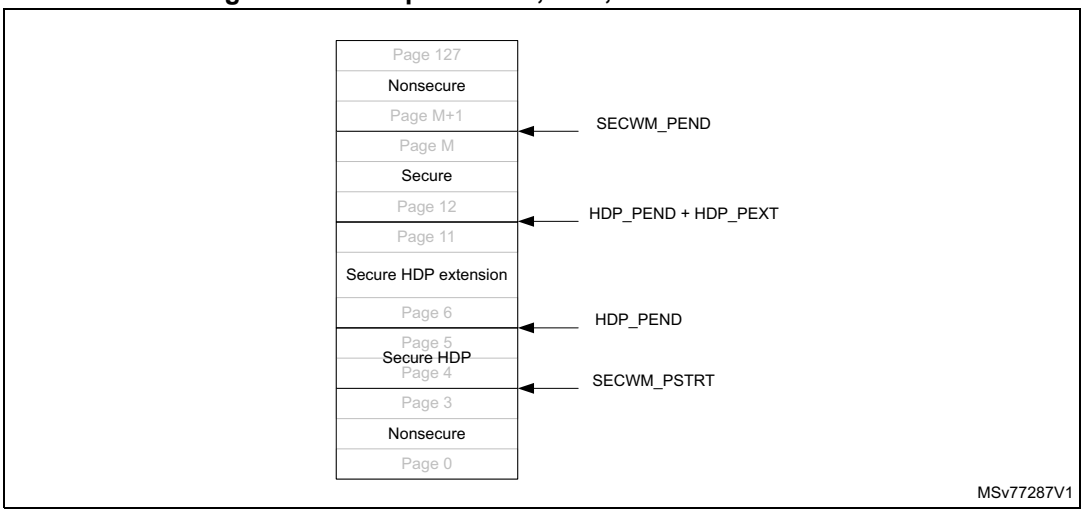
HDP and secure watermark option bytes values		Hide protection extension area
HDPEN = 0xB4	-	The secure HDP extension area is composed by the pages which follow SECWM_PSTRT ≤ pages < minimum of (SECWNM_PSTRT - 1 + HDP_PEXT and SECWM_PEND)
HDPEN different 0xB4	HDP_PEND < SECWM_PEND	The secure HDP extension is composed by the pages which follow HDP_PEND ≤ pages < minimum of (HDP_PEND + HDP_PEXT and SECWM_PEND)
	HDP_PEND = SECWM_PEND	No HDP extension area

For example, to protect by HDP from the address 0x0C00 4000 (included) to the address 0x0C00 5FFF (included), and HDP extension from 0x0C00 6000 to the address 0x0C00 BFFF:

- The option bytes registers must be programmed with:
  - SECWM_PSTRT = 0x4
  - HDP_PEND = 0x5
  - HDP_PEXT = 0x6
  - SECWM_PEND ≥ 0xB.

Pages from 4 to 5 (included) are protected by HDP and the pages from 6 to 11 (included) are protected by the HDP extension.

Figure 17. Example secure, HDP, HDP extension areas



7.5.5 Block-based secure flash memory area protection

Any page can be programmed on the fly as secure or nonsecure using the block-based configuration registers FLASH_SECBBRx to configure the security attribute.

When the page security attribute, bit SECBBy in SECBBRx, is set, the security attribute is the same as the secure watermark-based area. The secure page is only accessible by a secure access.

If the SECBBy bit in SECBBRx is set or reset for a page already included in a secure watermark-based area, the page keeps the watermark-based protection security attributes.

To modify a block-based page security attribution, the following actions are recommended:

- Check that no flash memory operation is ongoing on the related page.
- Add an ISB instruction after modifying the page security attribute bit in SECBBRx.

Caution: Switching a page or memory block from secure to nonsecure does not erase the content of the associated block. User secure software must perform the following needed operations before switching a block between secure and nonsecure attribute:

- Erase page content,
- Invalidate the instruction cache.

Note: For SECBBRx bit i access control, refer to Table 68.

7.5.6 Flash security attribute state

The flash memory is secure when at least one secure area is defined either by watermark-based option bytes or block-based security registers.

It is possible to override the flash security state using the INV bit in the FLASH_SECCR1 register.

The RCC FLASHEN and FLASHSMEN bits security attributes follow the flash memory security attribute. It is possible to override the flash memory security attribute in RCC using the INV bit in the FLASH_SECCR1 register. A secure firmware setting this INV bit allows a nonsecure firmware to disable the flash memory clock when the flash memory is in power down or when the MCU enters low-power modes.

Table 54. Flash security state

Secure area	INV bit	Flash security state
None	0	Nonsecure
	1	Secure
Yes	0	Secure
	1	Nonsecure

7.5.7 Block-based privileged flash memory area protection

Any page can be programmed on the fly as privileged or unprivileged using the block-based configuration registers FLASH_PRIVBBRx to configure the privilege attribute.

When the page privilege attribute, bit PRIVBBy in PRIVBBRx, is set, the page is only accessible by a privileged access. An unprivileged page is accessible by a privileged or unprivileged access.

To modify a block-based privilege attribution, the following actions are recommended:

- Check that no flash operation is ongoing on the related page.
- Add an ISB instruction after modifying the page security attribute bit in PRIVBBRx.

Caution: Switching a page or memory block from privileged to unprivileged does not erase the content of the associated block.

Note: For PRIVBBRx bit access control, refer to Table 69 and Table 70.

7.5.8 Flash memory registers privileged and unprivileged modes

The flash memory registers can be read and written by privileged and unprivileged accesses depending on the SPRIV and NSPRIV bits in *FLASH privilege configuration register (FLASH_PRIFCFGR)*, with the following rules:

- When the SPRIV (resp. NSPRIV) bit is reset, all secure (resp. nonsecure) flash memory registers can be read and written by both privileged or unprivileged access.
- When the SPRIV (resp. NSPRIV) bit is set, all secure (resp. nonsecure) flash memory registers can be read and written by privileged access only. Unprivileged access to a privileged register is RAZ/WI.

Table 67 summarizes the flash memory a privileged register access control.

7.6 Flash memory protection

The flash memory interface implements the following protection mechanisms:

- Readout protection (RDP)
- Two write protection (WRP) nonvolatile areas
- Additional secure protections when TrustZone is active (refer to *Section 7.5*)
  - One secure watermark-based nonvolatile area
  - One secure hide protection nonvolatile area
  - Secure block-based volatile areas with page granularity
- Privileged block-based volatile areas with page granularity (refer to *Section 7.5.7*)

7.6.1 Write protection (WRP)

The user area in flash memory can be protected against unwanted write operations. Two write-protected (WRP) areas can be defined, with page granularity.

Each area is defined by a start page offset and an end page offset related to the physical flash base address. These offsets are defined in the WRP address registers: *FLASH WRP area A address register (FLASH_WRPAR)*, and *FLASH WRP area B address register (FLASH_WRPBR)*.

The WRP “y” area (y = A,B) is defined as follows:

- From the address: flash base address + [WRPy_PSTRT x 0x2000] (included)
- To the address: flash base address + [(WRPy_PEND+1) x 0x2000] (excluded)

For example, to protect by WRP from the address 0x0806 2000 (included) to the address 0x0807 3FFF (included):

- FLASH_WRPAR register must be programmed with:
  - WRPA_PSTRT = 0x62
  - WRPA_PEND = 0x73

  WRPB_PSTRT and WRPB_PEND in FLASH_WRPBR can be used instead (area “B” in flash).

When WRP is active, protected flash memory pages cannot be erased or programmed. Consequently, a software mass erase cannot be performed if one area is write-protected.

If an erase/program operation to a write-protected part of the flash memory is attempted, the secure or nonsecure write protection error flag (WRPERR) is set in the FLASH_NSSR or

FLASH_SECSR register. This flag is also set for any write access to the system flash memory and to the OTP area.

Note: When the memory readout protection level 1 is selected (RDP level = 1), it is not possible to program or erase the flash memory (secure or nonsecure) if the CPU debug features are connected (JTAG or single wire) or boot code is being executed from RAM or system flash memory, even if WRP is not activated.

When the memory readout protection level 0.5 is selected (RDP level = 0.5), it is not possible to program or erase the flash secure memory if the CPU debug features are connected (JTAG or single wire), even if WRP is not activated.

Note: To validate the WRP options, the option bytes must be reloaded through the OBL_LAUNCH bit in the flash control register.

Table 55. WRP protection

WRP registers values (y = A / B)	WRP area
WRPy_PSTRT = WRPy_PEND	Page WRPy is protected
WRPy_PSTRT > WRPy_PEND	No WRPy area
WRPy_PSTRT < WRPy_PEND	The pages from WRPy_PSTRT to WRPy_PEND are protected

Write protection lock

Each WRP area can be independently locked by writing 0 to the UNLOCK bit in the *FLASH WRP area A address register (FLASH_WRPAR)*, or *FLASH WRP area B address register (FLASH_WRPBR)*. Once a WRP area is locked, it is not possible to modify its settings. In order to unlock a WRP area, a regression to RDP level 0 must be launched.

To make the WRP area immutable and act as a ROM, the following actions are needed:

- If RDP level is 0, 0.5 or 1, provision an OEM1 key in order to prevent a regression to RDP level 0 for users not knowing the key.
- If RDP level is 2, either provision an OEM1 key (refer to first bullet) or do not provision an OEM2 key (preventing regression from level 2 to level 1).

For more information on RDP regressions, refer to *Device life cycle managed by readout protection (RDP) transitions*.

7.6.2 Readout protection (RDP)

The readout protection protects the flash main memory, ICACHE, option bytes, backup registers, and the SRAMs.

Readout protection levels when Trustzone is disabled

There are three levels of readout protection from no protection (level 0) to maximum protection or no debug (level 2). The flash memory is protected according to the RDP option byte value shown in *Table 56*.

Table 56. Flash memory readout protection status (TZEN=0)

RDP byte value	Readout protection level
0xAA	Level 0
Any value except 0xAA or 0xCC	Level 1
0xCC	Level 2

- Level 0: no protection

  Read, program, and erase operations into the flash main memory area are possible. The option bytes, the SRAMs, and the backup registers are also accessible by all operations.
- Level 1: readout protection

  When the readout protection level 1 is set:
  - **User mode:** code executing in user mode (**boot flash**) can access the flash main memory, option bytes, SRAMs, and backup registers with all operations (read, erase, program).
  - **Debug, boot RAM, and bootloader modes**: in debug mode or when the microcontroller boots from RAM or system memory, the flash main memory, the backup registers, and the SRAM2 are totally inaccessible; any read or write access to the flash main memory generates a bus error and a hard fault interrupt. External memory on-the-fly decryption regions defined in OTFDEC are read as zero.
- Level 2: no debug

  When the readout protection level 2 is set:
  - The protection level 1 is guaranteed.
  - All debug features are disabled.
    - if OEM2 key has not been provided, JTAG and SWD are definitely disabled.
    - if OEM2 key has been provided, JTAG and SWD remain enabled under reset only to interface with DBGMCU_SR, DBGMCU_DBG_AUTH_HOST and DBG_MCU_AUTH_DEVICE registers to obtain device identification and provide OEM2 key to request RDP regression.
  - The boot from SRAM (boot RAM mode) and the boot from system memory (bootloader mode) are no longer available.
  - Only boot from the main flash memory is possible; all operations are allowed on the flash main memory. Read, erase, and program accesses to the flash memory and SRAMs from user code are allowed.
  - Option bytes cannot be programmed nor erased, any attempt to modify option bytes set the OPTWERR flag. Thus, the nonvolatile option areas can no longer be

modified, and level 2 cannot be removed: it is an irreversible operation unless an OEM2 key has been provisioned (refer to *OEM2 RDP lock mechanism*).

Note: The debug feature is also disabled under reset.

STMicroelectronics is not able to perform analysis on defective parts on which the level 2 protection has been set. Regress parts to RDP level 1 before returning them for analysis (refer to OEM2 RDP lock mechanism).

Table 57. Access status versus protection level and execution modes when TZEN = 0

Area	RDP level	User execution (boot from flash)			Debug/boot from RAM/ bootloader(1)		
		Read	Write	Erase	Read	Write	Erase
Flash main memory	1	Yes	Yes	Yes	No	No	No(4)
	2	Yes	Yes	Yes	N/A	N/A	N/A
System memory (2)	1	Yes	No	No	Yes	No	No
	2	Yes	No	No	N/A	N/A	N/A
Option bytes(3)	1	Yes	Yes(4)	N/A	Yes	Yes(4)	Yes
	2	Yes	No	N/A	N/A	N/A	N/A
OTP	1	Yes	Yes(5)	N/A	Yes	Yes(5)	N/A
	2	Yes	Yes(5)	N/A	N/A	N/A	N/A
Backup registers	1	Yes	Yes	N/A	No	No	N/A(6)
	2	Yes	Yes	N/A	N/A	N/A	N/A
SRAM2	1	Yes	Yes	N/A	No	No	N/A(6)
	2	Yes	Yes	N/A	N/A	N/A	N/A
OTFDEC regions	1	Yes	Yes	Yes	No	Yes	Yes(7)
	2	Yes	Yes	Yes	N/A	N/A	N/A

1. When the protection level 2 is active, the debug port, the boot from RAM and the boot from system memory are disabled.
2. The system memory is only read-accessible, whatever the protection level (0, 1 or 2) and execution mode.
3. Option bytes are only accessible through the flash registers interface and OPTSTRT bit.
4. The flash main memory is erased when the RDP option byte changes from level 1 to level 0.
5. OTP can only be written once.
6. All SRAMs and backup registers are erased when regressing RDP to level 0.
7. The OTFDEC keys are erased when regressing RDP to level 0.

Readout protection levels when Trustzone is enabled

There are four levels of readout protection from no protection (level 0) to maximum protection or no debug (level 2). The flash memory is protected according to the RDP option byte value shown in *Table 56*.

Table 58. Flash memory readout protection status (TZEN = 1)

RDP byte value	Readout protection level
0xAA	Level 0
0x55	Level 0.5
Any value except 0xAA or 0x55 or 0xCC	Level 1
0xCC	Level 2

- Level 0: no protection

  Read, program and erase operations into the flash main memory area are possible. The option bytes, the SRAMs, and the backup registers are also accessible by all operations.

  – **RSS mode**: when booting from RSS, the debug access is disabled while executing RSS code.

- Level 0.5: nonsecure debug only

  All read and write operations (if no write protection is set) from/to the nonsecure flash memory are possible. The debug access to the secure area is prohibited. Debug access to nonsecure area remains possible.

  – **User mode:** code executing in user mode (**boot flash**) can access the flash main memory, option bytes, SRAMs, and backup registers with all operations (read, erase, program).

  – **Nonsecure debug mode**: nonsecure debug is possible when the CPU is in nonsecure state. The secure flash memory, the secure backup registers and SRAMs area are inaccessible; the nonsecure flash memory, the nonsecure backup registers, and the nonsecure SRAMs area remain accessible for debug purpose.

  – **RSS mode**: when booting from RSS, the debug access is disabled while executing RSS code.

  – **Boot RAM mode**: boot from SRAM is not possible.

- Level 1: readout protection

  When the readout protection level 1 is set:

  – **User mode:** code executing in user mode (**boot flash**) can access the flash main memory, option bytes, SRAMs, and backup registers with all operations (read, erase, program).

  – **Nonsecure debug mode**: nonsecure debug is possible when the CPU is in nonsecure state. However, an intrusion is detected in case of debug access: the flash main memory, the backup registers, and the SRAM2 are totally inaccessible; any read or write access to the flash main memory generates a bus error and a

hard fault interrupt. External memory on-the-fly decryption regions defined in OTFDEC are read as zero.

  - **RSS mode**: when booting from RSS, the debug access is disabled while executing RSS code.
  - **Boot RAM mode**: boot from SRAM is not possible.

- Level 2: no debug

  When the readout protection level 2 is set:

  - The protection level 1 is guaranteed.
  - All debug features are disabled.
    - if OEM2 key has not been provided, JTAG and SWD are definitely disabled.
    - if OEM2 key has been provided, JTAG and SWD remain enabled under reset only to interface with DBGMCU_SR, DBGMCU_DBG_AUTH_HOST and DBG_MCU_AUTH_DEVICE registers to obtain device identification and provide OEM2 key to request RDP regression.
  - The boot from SRAM (boot RAM mode) and the boot from system memory (bootloader mode) are no longer available.
  - Boot from RSS is possible.
  - When booting from main flash or RSS, all operations are allowed on the flash main memory. Read, erase, and program accesses to flash memory and SRAMs from user code are allowed.
  - Option bytes cannot be programmed nor erased. Thus, the nonvolatile security areas SECWM, HDP, and WRP areas can no longer be modified, the level 2 cannot be removed: it is an irreversible operation unless an OEM2 key has been provisioned (refer to *OEM2 RDP lock mechanism*).

Note: The debug feature is also disabled under reset.

STMicroelectronics is not able to perform analysis on defective parts on which the level 2 protection has been set. Regress parts to RDP level 1 before returning them for analysis (refer to OEM2 RDP lock mechanism).

Table 59. Access status versus protection level and execution modes when TZEN = 1

Area	RDP level	User execution (boot from flash)			Debug/bootloader(1)		
		Read	Write	Erase	Read	Write	Erase
Flash main memory	0.5	Yes	Yes	Yes	Yes(2)	Yes(2)	Yes(2)
	1	Yes	Yes	Yes	No	No	No(5)
	2	Yes	Yes	Yes	N/A	N/A	N/A
System memory (3)	0.5	Yes	No	No	Yes	No	No
	1	Yes	No	No	Yes	No	No
	2	Yes	No	No	N/A	N/A	N/A
Option bytes(4)	0.5	Yes	Yes (5)	N/A	Yes	Yes (5)	Yes
	1	Yes	Yes(5)	N/A	Yes	Yes(5)	Yes
	2	Yes	No	N/A	N/A	N/A	N/A

Table 59. Access status versus protection level and execution modes when TZEN = 1 (continued)

Area	RDP level	User execution (boot from flash)			Debug/bootloader(1)		
		Read	Write	Erase	Read	Write	Erase
OTP	0.5	Yes	Yes(6)	N/A	Yes	Yes(6)	N/A
	1	Yes	Yes(6)	N/A	Yes	Yes(6)	N/A
	2	Yes	Yes(6)	N/A	N/A	N/A	N/A
Backup registers	0.5	Yes	Yes	N/A	Yes(2)	Yes(2)	N/A(7)
	1	Yes	Yes	N/A	No	No	N/A(7)
	2	Yes	Yes	N/A	N/A	N/A	N/A
SRAM2	0.5	Yes	Yes	N/A	Yes(2)	Yes(2)	N/A(7)
	1	Yes	Yes	N/A	No	No	N/A(7)
	2	Yes	Yes	N/A	N/A	N/A	N/A
OTFDEC regions	0.5	Yes	Yes	Yes	No	Yes	Yes(8)
	1	Yes	Yes	Yes	No	Yes	Yes(8)
	2	Yes	Yes	Yes	N/A	N/A	N/A

1. When the protection level 2 is active, the debug port and the bootloader mode are disabled.
2. Depends on TrustZone security access rights.
3. The system memory is only read-accessible, whatever the protection level (0, 1 or 2) and execution mode.
4. Option bytes are only accessible through the flash registers interface and OPTSTRT bit.
5. The flash main memory is erased when the RDP option byte regresses from level 1 to level 0.
6. OTP can only be written once.
7. All SRAMs and TAMP backup registers are erased when regressing RDP to level 0.5 and level 0.
8. The OTFDEC keys are erased when regressing RDP to level 0.5 and level 0

Device life cycle managed by readout protection (RDP) transitions

It is easy to move from level 0 or level 0.5 to level 1 by changing the value of the RDP byte to any value (except 0xCC, 0xAA, and 0x55). By programming the 0xCC value in the RDP byte, it is possible to go to level 2 either directly from level 0 or from level 0.5 or from level 1. Once in level 2, it is no longer possible to modify the readout protection level unless an OEM2 key is provisioned (refer to *OEM2 RDP lock mechanism*).

When the RDP is reprogrammed to the value 0xAA to move from level 1 to level 0, a mass erase of the flash main memory and SRAM1, SRAM2, ICACHE, PKA SRAM, TAMP backup registers, and OTFDEC keys is performed. The OTP area is not erased.

At RDP level 0.5, it is not possible to request RDP level 0. Instead, an RDP increase to level 1 followed by an RDP regression to level 0 is required.

When the RDP is programmed to the value 0x55 to move from level 1 to level 0.5, a partial mass erase of the flash main memory is performed. Only nonsecure watermark-based areas are erased (regardless if it is defined as secure by block-based). The SRAM1,

SRAM2, ICACHE, PKA SRAM, TAMP backup registers, and OTFDEC keys are mass erased. The OTP area is not erased. The RDP level 0.5 and partial nonsecure erase are only available when TrustZone is active.

Note: Full mass erase is performed only when level 1 is active and level 0 is requested. When the protection level is increased (0 to 0.5, 0 to 1, 0.5 to 1, 1 to 2, 0 to 2 or 0.5 to 2), there is no mass erase.

To validate the readout protection level change, the option bytes must be reloaded through the OBL_LAUNCH bit in FLASH control register (FLASH_NSCR1).

Before launching an RDP regression, the software must invalidate the ICACHE and wait for the BUSYF bit to get cleared.

Figure 18. RDP level transition scheme when TrustZone is disabled (TZEN = 0)

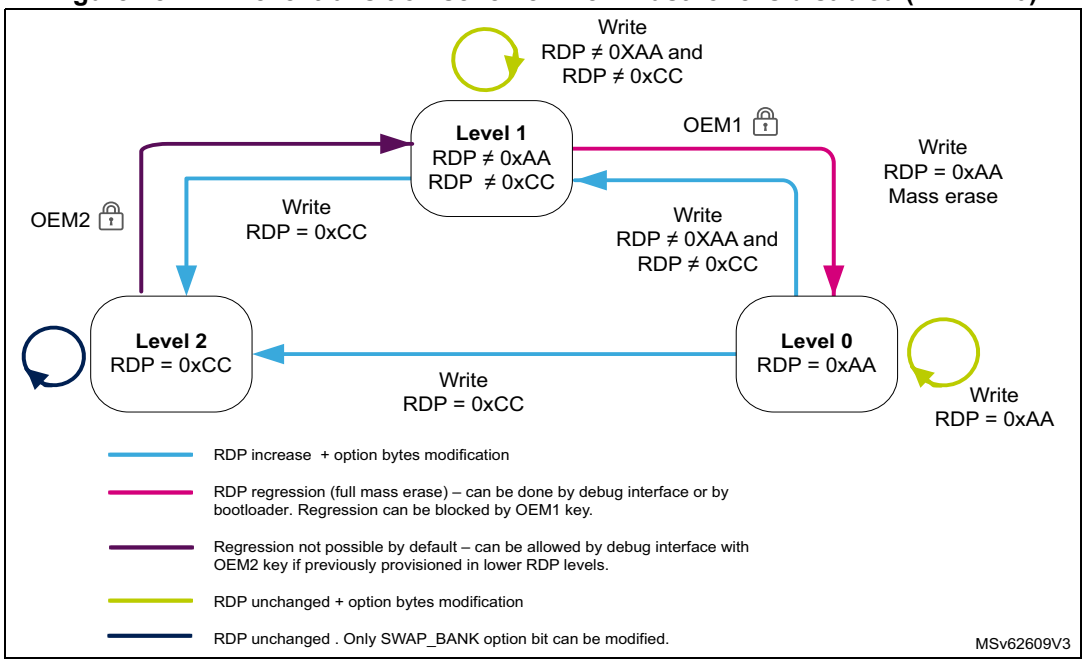
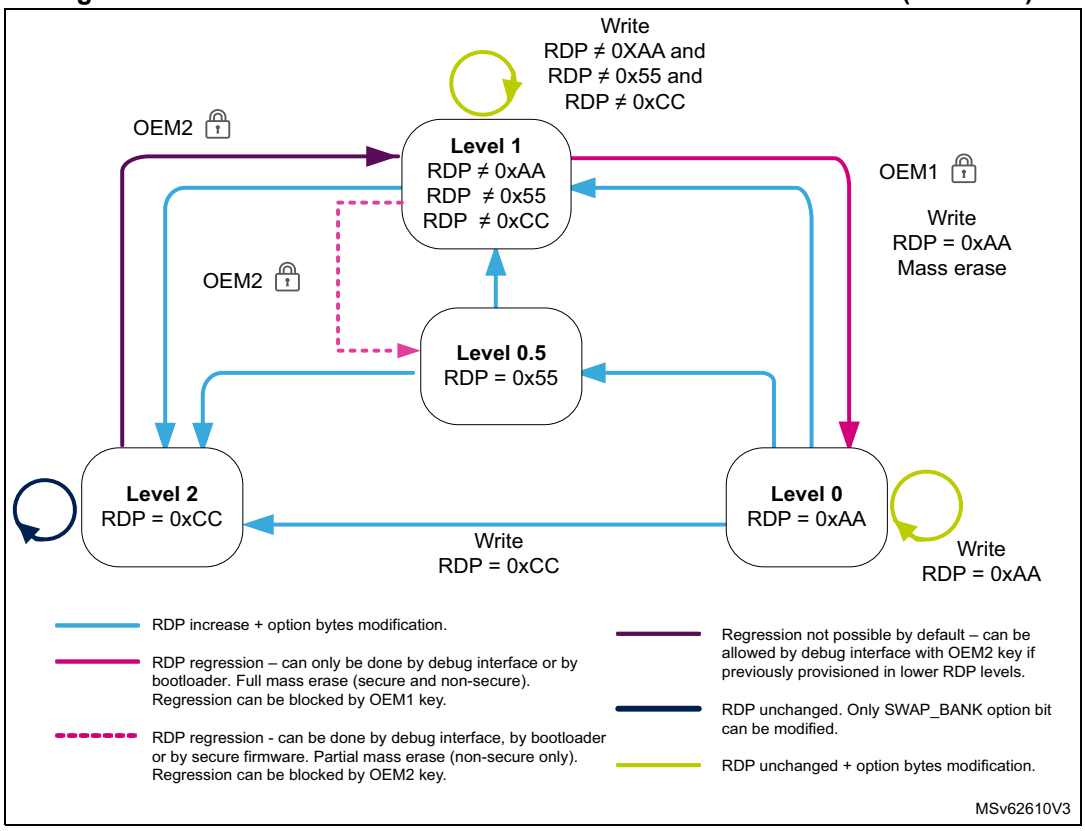


Figure 19. RDP level transition scheme when TrustZone is enabled (TZEN = 1)



OEM1/OEM2 lock activation

Two 128-bit keys (OEM1 key and OEM2 key) can be defined in order to lock the RDP regression. Each 128-bit key is coded on four registers:

1. OEM1 key coded on:
   a) *FLASH OEM1 key register 1 (FLASH_OEM1KEYR1)*
   b) *FLASH OEM1 key register 2 (FLASH_OEM1KEYR2)*
   c) *FLASH OEM1 key register 3 (FLASH_OEM1KEYR3)*
   d) *FLASH OEM1 key register 4 (FLASH_OEM1KEYR4)*
2. OEM2 key coded on:
   a) *FLASH OEM2 key register 1 (FLASH_OEM2KEYR1)*
   a) *FLASH OEM2 key register 2 (FLASH_OEM2KEYR2)*
   a) *FLASH OEM2 key register 3 (FLASH_OEM2KEYR3)*
   a) *FLASH OEM2 key register 4 (FLASH_OEM2KEYR4)*

OEM1 key and OEM2 key cannot be read through these registers but there are two 8-bit CRC OEM1KEYCRC and OEM2KEYCRC values which can be used as a public Key information.

Any read access to OEM1 key and OEM2 key returns zero.

OEM1 key can be modified in readout protection:

- RDP level 0
- RDP level 0.5 or 1 if OEM1LOCK = 0 in *FLASH status register (FLASH_NSSR)*

OEM2 key can be modified in readout protection:

- RDP level 0 or 0.5
- RDP level 1 if OEM2LOCK = 0 in *FLASH status register (FLASH_NSSR)*

When attempting to modify the OEM1 key registers or OEM2 key registers without following these rules, the user option modification is not done and the OPTWERR bit is set.

In order to activate the OEM1 key lock mechanism, the following steps are needed:

1. Check that the OEM1LOCK bit is not set or that the readout protection is at level 0.
2. Write a 128-bit OEM1 key in OEM1 key registers
3. Check key CRC in OEM1KEYCRC.
4. Launch option modification by setting the OPTSTRT bit in the *FLASH control register (FLASH_NSCR1)*.
5. Wait for the BSY bit to be cleared and check that OPTWERR is not set.
6. Set the OBL_LAUNCH option bit to start option bytes loading or perform a power-on reset.
7. Check that OEM1LOCK is set.

In order to activate the OEM2 key lock mechanism, the following steps are needed:

1. Check that the OEM2LOCK bit is not set or that the readout protection is at level 0 or 0.5.
2. Write a 128-bit OEM 2 key in OEM2 key registers.
3. Check key CRC in OEM2KEYCRC.
4. Launch option modification by setting the OPTSTRT bit in the *FLASH control register (FLASH_NSCR1)*.
5. Wait for the BSY bit to be cleared and check that OPTWERR is not set.
6. Set the OBL_LAUNCH option bit to start option bytes loading or perform a power-on reset.
7. Check that OEM2LOCK is set.

Note: The OEM1 key and OEM2 key must not contain more than four zero.

OEM1 RDP lock mechanism

The OEM1 key RDP lock mechanism is active when the OEM1LOCK bit is set. It conditions the RDP level 1 to RDP level 0 regression.

In order to regress from RDP level 1 to RDP level 0, the following sequence must be applied:

- Shift OEM1 key bit[31:0] followed by subsequent OEM1 keys and last OEM1 key bit[127:96] through the JTAG or SWD under reset in the DBGMCU_DBG_AUTH_HOST register.
  - If this key is matching the OEM1KEY register value, the RDP regression can be launched by setting the OPTSTRT bit.
  - If the key is not matching the OEM1KEY register value, RDP regression and any access to the flash are blocked until a next power on rest.

Attempting to regress from RDP level 1 to RDP level 0 without following this sequence sets the OPTWERR flag and the option bytes remain unchanged.

When the lock mechanism is not activated (OEM1LOCK =0), the regression from RDP level 1 to RDP level 0 is always granted.

OEM2 RDP lock mechanism

The OEM2 key RDP lock mechanism is active when the OEM2LOCK bit is set. It allows the following actions:

- Condition RDP level 1 to RDP level 0.5 regression.
- Authorize RDP level 2 to RDP level 1 regression.

In order to regress from RDP level 1 to RDP level 0.5, the following unlock sequence must be applied:

- Shift OEM2 key bits[31:0] followed by subsequent OEM2 keys and last OEM2 key bits[127:96] through the JTAG or SWD under reset in the DBGMCU_DBG_AUTH_HOST register.
  - If this key is matching the OEM2KEY register value, the RDP regression can be launched by setting the OPTSTRT bit.
  - If the key is not matching the OEM2KEY register value, RDP regression and any access to the flash are blocked until a next power on rest.

Attempting to regress from RDP level 1 to RDP level 0.5 without following this sequence, sets the OPTWERR flag and the option bytes remain unchanged.

To regress from RDP level 2 to RDP level 1, apply the following unlock sequence:

- Shift OEM2 key bits[31:0] followed by subsequent OEM2 keys and last OEM2 key bits[127:96] through the JTAG or SWD under reset in the DBGMCU_DBG_AUTH_HOST register.
  - If this key is matching the OEM2KEY register value, the RDP regression is launched by hardware (it is not possible to execute instructions when the key is matching). Subsequently apply a power-on reset (cycle V_{DD} power supply OFF and ON).
  - If the key is not matching the OEM2KEY register value, RDP regression and any access to the flash are blocked until a next power on rest.

Attempting to regress from RDP level 2 to RDP level 1 without following this sequence, leaves option bytes unchanged.

When the lock mechanism is not activated (OEM2LOCK =0), the following happens:

- The regression from RDP level 1 to RDP level 0.5 is always granted.
- The regression from RDP level 2 to RDP level 1 is never granted. When attempting to modify the options bytes, the OPTWERR flag is set and the option bytes remain unchanged.

OEM key CRC source

Below is the CRC source code that can be used to compare with the result of the computation provided by the flash interface in OEM1KEYCRC and OEM2KEYCRC fields, after loading the keys in OEM1 key and OEM2 key registers.

```
uint8_t getCRC(uint32_t * keyin)
{
const uint8_t CRC7_POLY = 0x7;
const uint32_t key_strobe[4] = {0xAA55AA55, 0x3, 0x18, 0xC0};
```

```
uint8_t i, j, k, crc = 0x0;
uint32_t keyval;
for (j = 0; j < 4; j++)
{keyval = * (keyin+j);
if (j == 0) { keyval ^= key_strobe[0]; }
else { keyval ^= (key_strobe[j] << 24) | (crc << 16) | (key_strobe[j] << 8)
| crc; }
for (i = 0, crc = 0; i < 32; i++)
{
k = (((crc >> 7) ^ (keyc=val >> (31-i))&0xF)) & 1;
crc <<= 1;
if (k) { crc ^= CRC7_POLY; }
}
crc^=0x55;
}
return crc;
}
```

7.7 Summary of flash memory and registers access control

The following tables summarize the flash memory and registers accesses status versus RDP level, WRP, and HDP protections.

Table 60. Flash memory access versus RDP level when TrustZone is active (TZEN = 1)

<table>
<tr><th colspan="2" rowspan="3">Access type</th><th colspan="3">RDP level 0, RDP level 0.5, RDP level 1 no intrusion(1) or RDP level 2</th><th>RDP level 1 with intrusion(2)</th></tr>
<tr><th rowspan="2">Nonsecure page</th><th colspan="2">Secure page</th><th rowspan="2">Nonsecure or secure page</th></tr>
<tr><th>HDP and HDP extension areas, ACCDIS disabled</th><th>Others(3)</th></tr>
<tr><td rowspan="4">Secure</td><td>Fetch</td><td>Bus error</td><td rowspan="2">RAZ</td><td rowspan="2">OK</td><td rowspan="3">Bus error</td></tr>
<tr><td>Read</td><td>RAZ, flash illegal access event</td></tr>
<tr><td>Write</td><td rowspan="2">WI, secure WRPERR flag set, flash illegal access event</td><td rowspan="2">WI, secure WRPERR flag set</td><td rowspan="2">No WRP: OK
WRP pages: WI and secure WRPERR flag set</td></tr>
<tr><td>Page erase</td><td>WI, secure WRPERR flag set</td></tr>
<tr><td rowspan="4">Nonsecure</td><td>Fetch</td><td rowspan="2">OK</td><td colspan="2">Bus error</td><td rowspan="2">Bus error</td></tr>
<tr><td>Read</td><td colspan="2">RAZ, flash illegal access event</td></tr>
<tr><td>Write</td><td>No WRP: OK
WRP pages: WI and nonsecure WRPERR flag set</td><td colspan="2" rowspan="2">WI, nonsecure WRPERR flag set, flash illegal access event</td><td rowspan="2">WI, nonsecure WRPERR flag set</td></tr>
<tr><td>Page erase</td><td>WI, nonsecure WRPERR flag set</td></tr>
</table>

1. RDP level 1 no intrusion: when booting from user flash memory and no debug access.
2. RDP level 1 with intrusion: when debug access detected.

3. Refers to flash memory secure configurations different from the one described for HDP protections. Example: flash memory secure and HDP area enabled but ACCDIS = enabled.

Table 61. Flash memory secure access versus HDP and HDP extension (TZEN = 1)

Access type	HDP area	HDP extension area	Other
Fetch	OK when HDP access is enabled. Else RAZ	OK when HDP area access is disabled and HDP extension access is enabled. Else RAZ	OK when HDP and HDP extension access are disabled. Else RAZ
Read	OK when HDP access is enabled. Else RAZ	OK when HDP extension access is enabled. Else RAZ	OK
Write	OK if HDP access is enabled. Else WI, secure WRPERR flag set	OK when HDP extension access is enabled. Else WI, secure WRPERR flag set	OK
Page erase			

Table 62. Flash memory access versus RDP level when TrustZone is disabled (TZEN = 0)

Access type	RDP level 0, RDP level1 no intrusion(1) or RDP level 2	RDP level 1 with intrusion(2)
Fetch	OK	Bus error
Read		
Write	No WRP: OK WRP pages: WI and nonsecure WRPERR flag set	
Erase		WI and nonsecure WRPERR flag set

1. RDP Level 1 no intrusion: when booting from user flash memory and no debug access.
2. RDP Level 1 with intrusion: when booting from RAM or system memory or debug access detected.

Table 63. Flash memory mass erase versus RDP level when TrustZone is active (TZEN = 1)

Access type		RDP level 0, RDP level 0.5, RDP level 1 no intrusion(1) or RDP level 2				RDP level 1 with intrusion(2)
		Nonsecure flash	Secure flash		Mix nonsecure and secure flash memory	Nonsecure or secure flash memory
			HDP area (HDPEN disable and ACCDIS disable)	Others(3)		
Secure	Mass erase	WI, secure WRPERR flag set, flash memory illegal access event	WI, secure WRPERR flag set	No WRP: OK WRP pages: WI and secure WRPERR flag set	WI, secure WRPERR flag set, flash memory illegal access event	WI, secure WRPERR flag set
Nonsecure	Mass erase	No WRP: OK WRP pages: WI and nonsecure WRPERR flag set	WI, nonsecure WRPERR flag set, flash memory illegal access event			WI, nonsecure WRPERR flag set

1. RDP Level 1 no intrusion: when booting from user flash memory and no debug access.
2. RDP Level 1 with intrusion: when debug access detected.
3. “Others” refers to flash memory secure configurations different from the one described for HDP protections. Example: flash memory secure and HDP area enabled but ACCDIS = enabled.

Table 64. Flash system memory, OTP and RSS accesses(1)

Access type		System memory (bootloader)	OTP(2)	RSS
Secure (TZEN = 1)	Fetch	Bus error		RAZ
	Read	RAZ, flash memory register illegal access event	OK	
	Write	WI, secure WRPERR flag set, flash memory illegal access event		WI, secure WRPERR flag set
Nonsecure (TZEN = 0 or TZEN = 1)	Fetch	OK	Bus error	Bus error
	Read		OK	RAZ(3)
	Write	WI and nonsecure WRPERR flag set	OK if not virgin: WI, nonsecure PROGERR flag set	WI, nonsecure WRPERR flag set

1. Valid for all RDP levels.
2. Write to a nonvirgin OTP generates a PGSERR.
3. Flash memory illegal access event is generated when TZEN = 1.

Table 65. Flash page access versus privilege mode(1)

Access type		Unprivileged page	Privileged page
Fetch, Read/Write, Page erase	Privileged	OK	
Fetch, Read	Unprivileged	OK	RAZ
Write, Page erase	Unprivileged		WI, secure or nonsecure WRPERR flag set

1. When TZEN = 1, access must be granted by security firewall before privilege is considered.

Table 66. Flash mass erase versus privilege mode(1)

Access type		Unprivileged flash memory	Privileged flash memory	Mix unprivileged and privileged flash memory
Mass erase	Privileged	OK		
Mass erase	Unprivileged	OK	WI, secure or nonsecure WRPERR flag set	

1. When TZEN = 1, access must be granted by security firewall before privilege is considered.

Table 67. Flash registers access(1)

Access type			Nonsecure register		Secure register	
			NSPRIV = 1	NSPRIV = 0	SPRIV = 1	SPRIV = 0
Fetch	Secure/ nonsecure	Privileged/ unprivileged	Bus error			
Read/ Write	Secure(2)	Privileged	OK			
		Unprivileged	RAZ/WI	OK	RAZ/WI	OK
	Nonsecure (3)	Privileged	OK		RAZ/WI and a flash memory register illegal access event(4)	
		Unprivileged	RAZ/WI	OK		

1. Except SECBBRx, PRIVBBRx and PRIVCFGR registers.
2. Secure access is valid only when TrustZone is active (TZEN = 1).
3. Nonsecure accesses are valid when TrustZone is active or disabled.
4. Flash register illegal access event is generated only when TZEN = 1.

Table 68. SECBBRx registers access when TrustZone is active (TZEN = 1)

Access type			Bit y in PRIVBBRx	Bit y in SECBBRx
Fetch	Secure/nonsecure	Privileged/unprivileged	-	Bus error
Read	Secure/nonsecure	Privileged/unprivileged	-	OK

Table 68. SECBBRx registers access when TrustZone is active (TZEN = 1) (continued)

Access type			Bit y in PRIVBBRx	Bit y in SECBBRx
Write	Secure	Privileged	x	OK
		Unprivileged	0	OK for bit y
		Unprivileged	1	WI for bit y
	Nonsecure	Privileged/unprivileged	x	WI and a flash memory register illegal access event

Table 69. PRIVBBRx registers access when TrustZone is active (TZEN = 1)

Access type			Page secure state (watermark or blocked based)	Bit y in PRIVBBRx
Fetch	Privileged/unprivileged	Secure/nonsecure	-	Bus error
Read	Privileged/unprivileged	Secure/nonsecure	-	OK for all bits
Write	Privileged	Secured	-	OK for all bits
		Nonsecure	Nonsecure	OK for bit y
		Nonsecure	Secure	WI for bit y
	Unprivileged	Secured/nonsecure	-	WI for all bits

Table 70. PRIVBBRx registers access when TrustZone is disabled (TZEN = 0)

Access type		PRIVBBRx
Fetch	Privileged/unprivileged	Bus error
Read	Privileged/unprivileged	OK
Write	Privileged	OK
	Unprivileged	WI

7.8 FLASH interrupts

Table 71. Flash interrupt requests

Interrupt vector	Interrupt event	Event flag	Event flag/interrupt clearing method	Interrupt enable control bit	Exit Sleep mode	Exit Stop and Standby modes
FLASH_S	Secure end of operation	Secure EOP(1)	Write secure EOP = 1	Secure EOPIE	Yes	No
	Secure operation error	Secure OPERR(2)	Write secure OPERR = 1	Secure ERRIE	Yes	No

Table 71. Flash interrupt requests (continued)

Interrupt vector	Interrupt event	Event flag	Event flag/interrupt clearing method	Interrupt enable control bit	Exit Sleep mode	Exit Stop and Standby modes
FLASH	Nonsecure end of operation	Nonsecure EOP(1)	Write nonsecure EOP = 1	Nonsecure EOPIE	Yes	No
	Nonsecure operation error	Nonsecure OPERR(2)	Write nonsecure OPERR = 1	Nonsecure ERRIE	Yes	No
	ECC correction	ECCC	Write ECCC=1	ECCCIE	Yes	No

1. Secure EOP (resp. nonsecure EOP) is set only if secure EOPIE (resp. nonsecure EOPIE) is set.
2. Secure OPERR (resp. nonsecure OPERR) is set only if secure ERRIE (resp. nonsecure ERRIE) is set.

7.9 FLASH registers

7.9.1 FLASH access control register (FLASH_ACR)

Address offset: 0x000

Reset value: 0x0000 0001

Access: no wait state when no flash memory read is ongoing; word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SLEEP_PD	Res.	PDREQ	LPM	Res.	Res.	PRFTEN	Res.	Res.	Res.	Res.	LATENCY[3:0]			
	rw		rs	rw			rw					rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **SLEEP_PD**: Flash memory power-down mode during Sleep mode

This bit determines whether the flash memory is in power-down mode or Idle mode when the device is in Sleep mode.

Access to the bit can be secured by PWR LPMSEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with FLASH SPRIV or when nonsecure with FLASH NSPRIV.

0: Flash in idle mode during Sleep mode

1: Flash in power-down mode during Sleep mode

Caution: The flash must not be put in power-down while a program or an erase operation is ongoing.

Caution: DMA can-not access the flash when in power-down mode. The DMA access to flash will be staled until the flash is taken out of power-down mode by a CPU wake-up. When the DMA needs to access flash while the CPU is in Sleep mode, SLEEP_PD must be 0.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PDREQ**: Flash power-down mode request

This bit requests flash to enter power-down mode. When flash enters power-down mode, this bit is cleared by hardware and the PDKEYR is locked.

This bit is write-protected with FLASH_PDKEYR.

Access to the bit can be secured by PWR LPMSEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with FLASH SPRIV or when nonsecure with FLASH NSPRIV.

0: No request for flash to enter power-down mode

1: Flash requested to enter power-down mode

Bit 11 **LPM**: Low-power read mode

This bit puts the flash memory in low-power read mode.

Access to the bit can be secured by PWR LPMSEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with FLASH SPRIV or when nonsecure with FLASH NSPRIV.

This bit cannot be written when a flash program or erase operation is busy (BSY = 1), or when the write buffer is not empty (WDW = 1). Changing this bit while a flash program or erase operation is busy (BSY = 1) is rejected.

0: Flash not in low-power read mode

1: Flash in low-power read mode

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **PRFTEN**: Prefetch enable

This bit enables the prefetch buffer in the embedded flash memory.

This bit can be protected against unprivileged access by FLASH NSPRIV.

0: Prefetch disabled

1: Prefetch enabled

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **LATENCY[3:0]**: Latency

These bits represent the ratio between the AHB hclk1 clock period and the flash memory access time.

Access to the bit can be secured by RCC SYSCLKSEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with FLASH SPRIV or when nonsecure with FLASH NSPRIV.

0000: Zero wait state

0001: One wait state

0010: Two wait states

...

1111: Fifteen wait states

Note: Before entering Stop 1, 2, and 3 mode software must set FLASH wait state latency to at least 1.

7.9.2 FLASH key register (FLASH_NSKEYR)

Address offset: 0x008

Reset value: 0x0000 0000

Access: one wait state; word access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV =1 in the FLASH_PRIVCFGR register.

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

Bits 31:0 **NSKEY[31:0]**: Flash memory nonsecure key

The following values must be written consecutively to unlock the FLASH_NSCR1 register, allowing the flash memory nonsecure programming/erasing operations:
KEY1: 0x4567 0123
KEY2: 0xCDEF 89AB

7.9.3 FLASH secure key register (FLASH_SECKEYR)

Address offset: 0x00C

Reset value: 0x0000 0000

Access: one wait state; word access

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

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

Bits 31:0 **SECKEY[31:0]**: Flash memory secure key

The following values must be written consecutively to unlock the FLASH_SECCR1 register, allowing the flash memory secure programming/erasing operations:
KEY1: 0x4567 0123
KEY2: 0xCDEF 89AB

7.9.4 FLASH option key register (FLASH_OPTKEYR)

Address offset: 0x010

Reset value: 0x0000 0000

Access: one wait state; word access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

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

Bits 31:0 **OPTKEY[31:0]**: Option byte key

The LOCK bit in the FLASH_NSCR1 must be cleared before doing the unlock sequence for OPTLOCK bit. The following values must be written consecutively to unlock the FLASH_NSCR1.OPTSTRT and OBL_LAUNCH register bits concerning user option operations:
KEY1: 0x0819 2A3B
KEY2: 0x4C5D 6E7F

7.9.5 FLASH power-down key register (FLASH_PDKEYR)

Address offset: 0x018

Reset value: 0x0000 0000

Access: no wait state; word access

Access to this register can be protected by PWR LPMSEC. When secure it can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when secure by SPRIV = 1 and when nonsecure by NSPRIV in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PDKEY1[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDKEY1[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **PDKEY1[31:0]**: Flash power-down key

The following values must be written consecutively to unlock the PDREQ bit in FLASH_ACR:
PDKEY_1: 0x0415 2637
PDKEY_2: 0xFAFB FCFD

7.9.6 FLASH status register (FLASH_NSSR)

Address offset: 0x020

Reset value: 0x000X 0000

Access: no wait state; word, half-word, and byte access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	OPBRK	Res.	Res.	Res.	Res.	Res.	PD	OEM2LOCK	OEM1LOCK	WDW	BSY
					rc_w1						r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OPTWERR	Res.	Res.	Res.	Res.	Res.	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res.	OPERR	EOP
		rc_w1						rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **OPBRK**: Nonsecure flash operation aborted

This bit indicates that a flash nonsecure operation is aborted. It is cleared by writing 1.
Flash operation abort information in FLASH_OPSR shall be read when OPBRK is set. Clearing OPBRK will also clear FLASH_OPSR.

Bits 25:21 Reserved, must be kept at reset value.

Bit 20 **PD**: Flash in power-down mode

This bit indicates that the flash memory is in power-down state. It is reset when flash is in normal mode or being awaken.

Bit 19 **OEM2LOCK**: OEM2 key RDP lock

This bit indicates that the OEM2 key read during the OBL is not virgin. When set, the OEM2 key RDP lock mechanism is active.

Bit 18 **OEM1LOCK**: OEM1 key RDP lock

This bit indicates that the OEM1 key read during the OBL is not virgin. When set, the OEM1 key RDP lock mechanism is active.

Bit 17 **WDW**: Nonsecure wait data to write

This bit indicates that the flash memory write buffer has been written by a secure or nonsecure operation. It is set when the first data is stored in the buffer and cleared when the write is performed in the flash memory.

Bit 16 **BSY**: Nonsecure busy

This indicates that a flash memory secure or nonsecure operation is in progress. This bit is set at the beginning of a flash operation and reset when the operation finishes or when an error occurs.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OPTWERR**: Option write error

This bit is set by hardware when the options bytes are written with an invalid configuration or when modifying options in RDP level 2. It is cleared by writing 1.
Refer to *Section 7.3.11* for full conditions of error flag setting.

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **PGSERR**: Nonsecure programming sequence error

This bit is set by hardware when programming sequence is not correct. It is cleared by writing 1.
Refer to *Section 7.3.11* for full conditions of error flag setting.

Bit 6 **SIZERR**: Nonsecure size error

This bit is set by hardware when the size of the access is a byte or half-word during a nonsecure program sequence. Only double-word programming is allowed by means of successive word accesses. This bit is cleared by writing 1.

Bit 5 **PGAERR**: Nonsecure programming alignment error

This bit is set by hardware when the first word to be programmed is not aligned with a double-word address, or the second word does not belong to the same double-word address. This bit is cleared by writing 1.

Bit 4 **WRPERR**: Nonsecure write protection error

This bit is set by hardware when a nonsecure address to be erased/programmed belongs to a write-protected part (by WRP or is secure) of the flash memory. This bit is cleared by writing 1.
Refer to *Section 7.3.11* for full conditions of error flag setting.

Bit 3 **PROGERR**: Nonsecure programming error

This bit is set by hardware when a nonsecure double-word address to be programmed contains a value different from all 1 before programming, except if the data to write is all 0. This bit is cleared by writing 1.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **OPERR**: Nonsecure operation error

This bit is set by hardware when a flash memory nonsecure operation (program/erase) completes unsuccessfully. This bit is set only if nonsecure error interrupts are enabled (NSERRIE = 1). This bit is cleared by writing 1.

Bit 0 **EOP**: Nonsecure end of operation

This bit is set by hardware when one or more flash memory nonsecure operation (program/erase) has been completed successfully. This bit is set only if the nonsecure end of operation interrupts are enabled (EOPIE = 1 in FLASH_NSCR1). This bit is cleared by writing 1.

7.9.7 FLASH secure status register (FLASH_SECSR)

Address offset: 0x024

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	OPBRK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDW	BSY
					rc_w1									r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res.	OPERR	EOP
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **OPBRK**: Secure flash operation aborted

This bit indicates that a flash secure operation is aborted. It is cleared by writing 1.
Flash operation abort information in FLASH_OPSR shall be read when OPBRK is set.
Clearing OPBRK will also clear FLASH_OPSR.

Bits 25:18 Reserved, must be kept at reset value.

Bit 17 **WDW**: Secure wait data to write

This bit indicates that the flash memory write buffer has been written by a secure or nonsecure operation. It is set when the first data is stored in the buffer and cleared when the write is performed in the flash memory.

Bit 16 **BSY**: Secure busy

This bit indicates that a flash memory secure or nonsecure operation is in progress. This is set on the beginning of a flash operation and reset when the operation finishes or when an error occurs.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **PGSERR**: Secure programming sequence error

This bit is set by hardware when programming sequence is not correct. It is cleared by writing 1.

Refer to *Section 7.3.11: Flash memory errors flags* for full conditions of error flag setting.

Bit 6 **SIZERR**: Secure size error

This bit is set by hardware when the size of the access is a byte or half-word during a secure program sequence. Only double-word programming is allowed by means of successive word accesses.This bit is cleared by writing 1.

Bit 5 **PGAERR**: Secure programming alignment error

This bit is set by hardware when the first word to be programmed is not aligned with a double-word address, or the second, third or forth word does not belong to the same double-word address.This bit is cleared by writing 1.

Bit 4 **WRPERR**: Secure write protection error

This bit is set by hardware when an secure address to be erased/programmed belongs to a write-protected part (by WRP, HDP or HDP extension) of the flash memory. This bit is cleared by writing 1.

Refer to *Section 7.3.11* for full conditions of error flag setting.

Bit 3 **PROGERR**: Secure programming error

This bit is set by hardware when a secure double-word address to be programmed contains a value different from all 1 before programming, except if the data to write is all 0. This bit is cleared by writing 1.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **OPERR**: Secure operation error

This bit is set by hardware when a flash memory secure operation (program/erase) completes unsuccessfully. This bit is set only if secure error interrupts are enabled (SECERRIE = 1). This bit is cleared by writing 1.

Bit 0 **EOP**: Secure end of operation

This bit is set by hardware when one or more flash memory secure operation (program/erase) has been completed successfully. This bit is set only if the secure end of operation interrupts are enabled (EOPIE = 1 in FLASH_SECCR1). This bit is cleared by writing 1.

7.9.8 FLASH control register (FLASH_NSCR1)

Address offset: 0x028

Reset value: 0xC000 0000

Access: no wait state when no flash memory operation is ongoing; word, half-word, and byte access

This register is write protected with LOCK and must be unlocked by the NSKEY unlock sequence.

This register can only be written when the BSY or OBL_LAUNCH are reset. Otherwise, the write access is ignored and generates a bus error and hard fault interrupt.

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OPTLOCK	Res.	Res.	OBL_LAUNCH	Res.	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	Res.	OPTSTRT	STRT
rs	rs			rc_w1		rw	rw							rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	BWR	Res.	Res.	Res.	Res.	PNB[6:0]							MER	PER	PG
	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK:** Nonsecure lock

This bit is set only.

When set, the FLASH_NSCR1 register write access is locked. This bit is cleared by hardware after detecting the unlock sequence in FLASH_NSKEYR.

In case of an unsuccessful unlock operation, this bit remains set until the next system reset.

Bit 30 **OPTLOCK:** Option lock

This bit is set only. When set, the FLASH_NSCR1.OPTSRT and OBL_LAUNCH bits concerning user options write access is locked. This bit is cleared by hardware after detecting the unlock sequence in FLASH_OPTKEYR. The FLASH_NSCR1.LOCK bit must be cleared before doing the FLASH_OPTKEYR unlock sequence.

In case of an unsuccessful unlock operation, this bit remains set until the next reset.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **OBL_LAUNCH**: Force the option byte loading

When set to 1, this bit forces the option byte reloading. This bit is cleared only when the option byte loading is complete. This bit is write-protected with OPTLOCK.

0: Option byte loading complete

1: Option byte loading requested

Note: The LSE oscillator must be disabled, LSEON = 0 and LSERDY = 0, before starting OBL_LAUNCH.

Bit 26 Reserved, must be kept at reset value.

Bit 25 **ERRIE**: Nonsecure error interrupt enable

This bit enables the interrupt generation when the OPERR bit in the FLASH_NSSR is set to 1.

0: Nonsecure OPERR error interrupt disabled

1: Nonsecure OPERR error interrupt enabled

Bit 24 **EOPIE**: Nonsecure end of operation interrupt enable
This bit enables the interrupt generation when the EOP bit in the FLASH_NSSR is set to 1.
0: Nonsecure EOP Interrupt disabled
1: Nonsecure EOP Interrupt enabled

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **OPTSTRT**: Options modification start
This bit triggers an option bytes erase and program operation when set. This bit is write-protected with OPTLOCK.. This bit is set only by software, and is cleared when the BSY bit is cleared in FLASH_NSSR.

Bit 16 **STRT**: Nonsecure operation start
This bit triggers a nonsecure erase operation when set. If MER and PER bits are reset and the STRT bit is set, the PGSERR bit in FLASH_NSSR is set (this condition is forbidden).
This bit is set only by software and is cleared when the BSY bit is cleared in FLASH_NSSR.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **BWR**: Nonsecure burst write programming mode
When set, this bit selects the burst write programming mode.

Bits 13:10 Reserved, must be kept at reset value.

Bits 9:3 **PNB[6:0]**: Nonsecure page number selection
These bits select the page to erase.
0000000: page 0
0000001: page 1
...
1111111: page 127

Bit 2 **MER**: Nonsecure flash mass erase
This bit triggers the flash nonsecure mass erase (all user pages) when set.

Bit 1 **PER**: Nonsecure page erase
0: Nonsecure page erase disabled
1: Nonsecure page erase enabled

Bit 0 **PG**: Nonsecure programming
0: Nonsecure flash programming disabled
1: Nonsecure flash programming enabled

7.9.9 FLASH secure control register (FLASH_SECCR1)

Address offset: 0x02C

Reset value: 0x8000 0000

Access: no wait state when no flash memory operation is ongoing; word, half-word, and byte access

Access to this register is locked and must be unlocked by the SECKEY unlock sequence.

This register can only be written when the BSY or OBL_LAUNCH are reset. Otherwise, the write access is ignored and generates a bus error and hard fault interrupt.

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	INV	Res.	Res.	Res.	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT
rs		rw				rw	rw								rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	BWR	Res.	Res.	Res.	Res.	PNB[6:0]							MER	PER	PG
	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK:** Secure lock

This bit is set only. When set, the FLASH_SECCR1 register is locked. It is cleared by hardware after detecting the unlock sequence in FLASH_SECKEYR register.

In case of an unsuccessful unlock operation, this bit remains set until the next system reset.

Bit 30 Reserved, must be kept at reset value.

Bit 29 **INV:** Flash memory security state invert

This bit inverts the flash memory security state.

Bits 28:26 Reserved, must be kept at reset value.

Bit 25 **ERRIE**: Secure error interrupt enable

This bit enables the interrupt generation when the OPERR bit in FLASH_SECSR is set to 1.

0: Secure OPERR error interrupt disabled

1: Secure OPERR error interrupt enabled

Bit 24 **EOPIE**: Secure End of operation interrupt enable

This bit enables the interrupt generation when the EOP bit in FLASH_SECSR is set to 1.

0: Secure EOP Interrupt disabled

1: Secure EOP Interrupt enabled

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **STRT**: Secure start

This bit triggers a secure erase operation when set. If MER and PER bits are reset and the STRT bit is set, the PGSERR in the FLASH_SECSR is set (this condition is forbidden).

This bit is set only by software and is cleared when the BSY bit is cleared in FLASH_SECSR.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **BWR**: Secure burst write programming mode

When set, this bit selects the burst write programming mode.

Bits 13:10 Reserved, must be kept at reset value.

Bits 9:3 **PNB[6:0]**: Secure page number selection
These bits select the page to erase:
0000000: page 0
0000001: page 1
...
1111111: page 127

Bit 2 **MER**: Secure flash mass erase
This bit triggers the flash secure mass erase (all user pages) when set.

Bit 1 **PER**: Secure page erase
0: Secure page erase disabled
1: Secure page erase enabled

Bit 0 **PG**: Secure programming
0: Secure flash programming disabled
1: Secure flash programming enabled

7.9.10 FLASH ECC register (FLASH_ECCR)

Address offset: 0x030

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECCD	ECCC	Res.	Res.	Res.	Res.	Res.	ECCIE	Res.	SYSF_ECC	Res.	Res.	Res.	ADDR_ECC[18:16]		
rc_w1	rc_w1						rw		r				r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_ECC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **ECCD**: ECC detection
This bit is set by hardware when two ECC errors have been detected (only if ECCC and ECCD were previously cleared). When this bit is set, a NMI is generated. This bit is cleared by writing 1.

Bit 30 **ECCC**: ECC correction
This bit is set by hardware when one ECC error has been detected and corrected (only if ECCC and ECCD were previously cleared). An interrupt is generated if ECCIE is set. This bit is cleared by writing 1.

Bits 29:25 Reserved, must be kept at reset value.

Bit 24 **ECCIE**: ECC correction interrupt enable
This bit enables the interrupt generation when the ECCC bit in the FLASH_ECCR register is set.
0: ECCC interrupt disabled
1: ECCC interrupt enabled.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **SYSF_ECC**: System flash memory ECC fail

This bit indicates that the ECC error correction or double ECC error detection is located in the system flash memory.

Bits 21:19 Reserved, must be kept at reset value.

Bits 18:0 **ADDR_ECC[18:0]**: ECC fail address

This field indicates which address is concerned by the ECC error correction or by the double ECC error detection. The address is given relative to flash base address, from offset 0x0 0000 to 0x7 FFF0.

7.9.11 FLASH operation status register (FLASH_OPSR)

Address offset: 0x034

Reset value: 0xXXXX XXXX

(0xXXXX XXXX after system reset and 0x0000 0000 after power-on reset)

Access: no wait state; word, half-word and byte access

This register is nonsecure. It can be read by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CODE_OP[2:0]			ET_OP[5:0]						SYSF_OP	Res.	Res.	Res.	ADDR_OP[18:16]		
r	r	r	r	r	r	r	r	r	r				r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_OP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 **CODE_OP[2:0]**: Flash memory operation code

This field indicates which flash memory operation has been interrupted:
000: No flash operation interrupted
001: Single write operation interrupted
010: Burst write operation interrupted
011: Page erase operation interrupted
100: Reserved
101: Mass erase operation interrupted
110: Option change operation interrupted
111: Reserved

Bits 28:23 **ET_OP[5:0]**: Elapsed erase time or burst programmed double-words before interruption

On an erase interruption, this field indicates the elapsed erase time before interruption with a 500 µs resolution.
On a burst programming interruption, this field indicates the number of correctly programed double-words before interruption.

Bit 22 **SYSF_OP**: Operation in system flash memory interrupted
This bit indicates that an interruption occurred during an operation in the system flash memory.

Bits 21:19 Reserved, must be kept at reset value.

Bits 18:0 **ADDR_OP[18:0]**: Interrupted operation address
This field indicates which address in the flash memory was accessed when an interruption occurred. The address is given relative to the flash base address, from offset 0x0 0000 to 0x7 FFF8.

7.9.12 FLASH control 2 register (FLASH_NSCR2)

Address offset: 0x038

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPA	Res.	Res.	Res.	Res.	Res.	Res.	ES	PS
							rs							rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **OPA**: Flash operation abort
This register bit can be set by software or hardware, and is cleared by hardware.
This register bit is software write protected with LOCK and must be unlocked by NSKEY unlock. Hardware can set this bit regardless LOCK.
0: no effect
1: writing 1 when a flash nonsecure operation is ongoing will abort the ongoing flash nonsecure operation.
Note: writing 1 when no flash operation is ongoing is ignored.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **ES**: Erase suspend request
0: erase suspend disabled
1: erase suspend requested (enabled)

Bit 0 **PS**: Program suspend request
0: program suspend disabled
1: program suspend requested (enabled)

7.9.13 FLASH secure control 2 register (FLASH_SECCR2)

Address offset: 0x03C

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPA	Res.	Res.	Res.	Res.	Res.	Res.	ES	PS
							rs							rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **OPA**: Flash operation abort

This register bit can be set by software or hardware, and is cleared by hardware.

This register bit is software write protected with LOCK and must be unlocked by NSKEY unlock. Hardware can set this bit regardless LOCK.

0: no effect

1: writing 1 when a flash secure or nonsecure operation is ongoing will abort the ongoing secure or nonsecure flash operation.

Note: writing 1 when no flash operation is ongoing is ignored.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **ES**: Erase suspend request

0: erase suspend disabled

1: erase suspend requested (enabled)

Bit 0 **PS**: Program suspend request

0: program suspend disabled

1: program suspend requested (enabled)

7.9.14 FLASH option register (FLASH_OPTR)

Address offset: 0x040

ST factory programmed value: 0x1FFF F8AA

Reset value: 0xXXXX XXXX

Register bits 0 to 31 are loaded with values from the flash memory at OBL.

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TZEN	Res.	Res.	Res.	NBOOT0	NSWBOOT0	SRAM2_RST	SRAM2_PE	Res.	Res.	Res.	Res.	Res.	IWDG_STDBY	IWDG_STOP	IWDG_SW
rw				rw	rw	rw	rw						rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRAM1_RST	Res.	NRST_STDBY	NRST_STOP	Res.	BOR_LEV[2:0]			RDP[7:0]							
rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TZEN:** Global TrustZone security enable
0: Global TrustZone security disabled
1: Global TrustZone security enabled

Bits 30:28 Reserved, must be kept at reset value.

Bit 27 **NBOOT0:** NBOOT0 option bit
0: NBOOT0 = 0
1: NBOOT0 = 1

Bit 26 **NSWBOOT0:** Software BOOT0
0: BOOT0 taken from the option bit NBOOT0
1: BOOT0 taken from PH3/BOOT0 pin

Bit 25 **SRAM2_RST:** SRAM2 erase upon system reset
0: SRAM2 erased when a system reset occurs
1: SRAM2 not erased when a system reset occurs

Bit 24 **SRAM2_PE:** SRAM2 parity check disable
0: SRAM2 parity check enabled
1: SRAM2 parity check disabled

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **IWDG_STDBY:** Independent watchdog counter freeze in Standby mode
0: Independent watchdog counter frozen in Standby mode
1: Independent watchdog counter running in Standby mode

Bit 17 **IWDG_STOP:** Independent watchdog counter freeze in Stop mode
0: Independent watchdog counter frozen in Stop mode
1: Independent watchdog counter running in Stop mode

Bit 16 **IWDG_SW:** Independent watchdog enable selection
0: Hardware mode, independent watchdog started automatically be hardware on reset selected
1: Software mode, independent watchdog started by software command selected

Bit 15 **SRAM1_RST:** SRAM1 erase upon system reset
0: SRAM1erased when a system reset occurs
1: SRAM1 not erased when a system reset occurs

Bit 14 Reserved, must be kept at reset value.

Bit 13 **NRST_STDBY:** Reset generation in Standby mode
0: Reset generated when entering the Standby mode
1: No reset generated when entering the Standby mode

Bit 12 **NRST_STOP:** Reset generation in Stop mode
0: Reset generated when entering the Stop mode
1: No reset generated when entering the Stop mode

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **BOR_LEV[2:0]:** BOR reset level
These bits contain the V_{DD} supply level threshold that activates/releases the reset.
000: BOR level 0 (reset level threshold around 1.7 V)
001: BOR level 1 (reset level threshold around 2.0 V)
010: BOR level 2 (reset level threshold around 2.2 V)
011: BOR level 3 (reset level threshold around 2.5 V)
100: BOR level 4 (reset level threshold around 2.8 V)

Bits 7:0 **RDP[7:0]:** Readout protection level
0xAA: Level 0 (readout protection not active)
0x55: Level 0.5 (readout protection not active, only nonsecure debug access is possible). Only available when TrustZone is active (TZEN=1)
0xCC: Level 2 (chip readout protection active)
Others: Level 1 (memories readout protection active)
Note: Refer to Section 7.6.2 for more details.

7.9.15 FLASH boot address 0 register (FLASH_NSBOOTADD0R)

Address offset: 0x044

ST factory programmed value: 0x0800 007F

Reset value: 0xXXXX XXXX

The option bytes are loaded with values from the flash memory at reset release.

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSBOOTADD0[24:9]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NSBOOTADD0[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw							

Bits 31:7 **NSBOOTADD0[24:0]**: Nonsecure boot base address 0
The nonsecure boot memory address can be programmed to any address in the valid address range (see *Table 8: Boot space versus RDP protection*) with a granularity of 128 bytes. These bits correspond to address [31:7]. The NSBOOTADD0 option bytes are selected following the BOOT0 pin or NSWBOOT0 state.
Examples:
NSBOOTADD0[24:0] = 0x0100000: Boot from flash memory (0x0800 0000)
NSBOOTADD0[24:0] = 0x017F100: Boot from system memory bootloader (0x0BF8 8000)
NSBOOTADD0[24:0] = 0x0400200: Boot from SRAM2 on S-Bus (0x2001 0000)

Bits 6:0 Reserved, must be kept at reset value.

7.9.16 FLASH boot address 1 register (FLASH_NSBOOTADD1R)

Address offset: 0x048

ST factory programmed value: 0x0BF8 F07F

Reset value: 0xXXXX XXXX

The option bytes are loaded with values from the flash memory at reset release.

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSBOOTADD1[24:9]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NSBOOTADD1[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw							

Bits 31:7 **NSBOOTADD1[24:0]**: Nonsecure boot address 1

The nonsecure boot memory address can be programmed to any address in the valid address range (see *Table 8: Boot space versus RDP protection*) with a granularity of 128 bytes. These bits correspond to address [31:7]. The NSBOOTADD0 option bytes are selected following the BOOT0 pin or NSWBOOT0 state.

Examples:

NSBOOTADD1[24:0] = 0x0100000: boot from flash memory (0x0800 0000)

NSBOOTADD1[24:0] = 0x017F100: boot from system memory bootloader (0x0BF8 8000)

NSBOOTADD1[24:0] = 0x0400200: boot from SRAM2 (0x2001 0000)

Bits 6:0 Reserved, must be kept at reset value.

7.9.17 FLASH secure boot address 0 register (FLASH_SECBOOTADD0R)

Address offset: 0x04C

ST factory programmed value: 0x0C00 007C

Reset value: 0xXXXX XXXX

The option bytes are loaded with values from the flash memory at reset release.

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SECBOOTADD0[24:9]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SECBOOTADD0[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	BOOT_LOCK
rw	rw	rw	rw	rw	rw	rw	rw	rw							rs

Bits 31:7 **SECBOOTADD0[24:0]**: Secure boot base address 0

This address is only used when TZEN = 1.

The secure boot memory address can be programmed to any address in the valid address range (see *Table 8: Boot space versus RDP protection*) with a granularity of 128 bytes. This bits correspond to address [31:7] The SECBOOTADD0 option bytes are selected following the BOOT0 pin or NSWBOOT0 state.

Examples:

SECBOOTADD0[24:0] = 0x018 0000: Boot from secure user flash memory (0x0C00 0000)

SECBOOTADD0[24:0] = 0x01F F000: Boot from RSS system flash memory (0x0FF8 0000)

SECBOOTADD0[24:0] = 0x060 0000: Boot from secure SRAM1 on S-Bus (0x3000 0000)

Bits 6:1 Reserved, must be kept at reset value.

Bit 0 **BOOT_LOCK**: Boot lock

This lock is only used when TZEN = 1.

When set, the boot is always forced to base address value programmed in SECBOOTADD0[24:0] option bytes whatever the boot selection option. When set, this bit can only be cleared by an RDP regression level 1 to level 0.

7.9.18 FLASH secure watermark register 1 (FLASH_SECWMR1)

Address offset: 0x050

ST factory programmed value: 0xFFFF FF80

Reset value: 0xXXXX XXXX

Register bits are loaded with values from the flash memory at OBL. Reserved bits are read as 1.

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM_PEND[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM_PSTRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **SECWM_PEND[6:0]:** End page of secure area
This field contains the last page of the secure area.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **SECWM_PSTRT[6:0]:** Start page of secure area
This field contains the first page of the secure area.

7.9.19 FLASH secure watermark register 2 (FLASH_SECWMR2)

Address offset: 0x054

ST factory programmed value: 0xB480 FFFF

Reset value: 0xXXXX XXXX

Register bits are loaded with values from the flash memory at OBL.

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

This register is secure, it can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HDPEN[7:0]								Res.	HDP_PEND[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:24 **HDPEN[7:0]**: Secure Hide protection area enable
0xB4: No secure HDP area
others: Secure HDP area enabled

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **HDP_PEND[6:0]**: End page of secure hide protection area
This field contains the last page of the secure HDP area.

Bits 15:0 Reserved, must be kept at reset value.

7.9.20 FLASH WRP area A address register (FLASH_WRPAR)

Address offset: 0x058

ST factory programmed value: 0xFF80 FFFF

Reset value: 0xXXXX XXXX

Register bits are loaded with values from the flash memory at OBL. Reserved bits are read as 1.

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPA_PEND[6:0]						
rw									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPA_PSTRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 **UNLOCK**: WPR area A unlock
0: WRPA start and end pages locked
1: WRPA start and end pages unlocked

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:16 **WRPA_PEND[6:0]**: WPR area A end page
This field contains the last page of the WPR area A.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **WRPA_PSTRT[6:0]**: WPR area A start page
This field contains the first page of the WPR area A.

7.9.21 FLASH WRP area B address register (FLASH_WRPBR)

Address offset: 0x05C

ST factory programmed value: 0xFF80 FFFF

Reset value: 0xXXXX XXXX

Register bits are loaded with values from the flash memory at OBL.

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPB_PEND[6:0]						
rw									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPB_PSTRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 **UNLOCK**: WPR area B unlock
0: WRPB start and end pages locked
1: WRPB start and end pages unlocked

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:16 **WRPB_PEND[6:0]**: WRP area B end page
This field contains the last page of the WRP area B.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **WRPB_PSTRT[6:0]**: WRP area B start page
This field contains the first page of the WRP area B.

7.9.22 FLASH secure block based register x (FLASH_SECBBRx)

Address offset: 0x080 + 0x4 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

This register is secure. It can be written only by secure access. Individual register bits can be protected against unprivileged access (refer to *Table 69*).

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

Bits 31:0 **SECBB[31:0]:** page secure/nonsecure attribution (y = 0 to 31)
Each bit is used to set one page security attribution.
0: Page (32 x (x - 1) + y) in flash not block-based secure
1: Page (32 x (x - 1) + y) in flash block-based secure

7.9.23 FLASH secure HDP control register (FLASH_SECHDPCR)

Address offset: 0x0C0

Reset value: 0x00A3 00A3

Access: no wait state; word, half-word, and byte access

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPEXT_ACCDIS[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP_ACCDIS[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **HDPEXT_ACCDIS[7:0]:** Secure HDP extension area access disable
When set to a value different than 0xA3, these bit can't be modified anymore, except by a system reset or standby reset.
0xA3: Access to secure HDP extension area granted
0x5C: Access to secure HDP area and HDP extension area denied, and HDP_PEXT increment allowed at any time. SECWMRx option bytes modification blocked.
other: Access to secure HDP area, HDP extension area, and HDP_PEXT denied. SECWMRx option bytes modification blocked.
(refer to *Rules for modifying specific option bytes*)

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **HDP_ACCDIS[7:0]:** Secure HDP area access disable
When set to a value different than 0xA3, these bit can't be modified anymore, except by a system reset or standby reset.
0xA3: Access to secure HDP area granted
other: Access to secure HDP area denied. SECWMRx option bytes modification blocked.
(refer to *Rules for modifying specific option bytes*)

7.9.24 FLASH privilege configuration register (FLASH_PRIFCFGR)

Address offset: 0x0C4

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

This register is privileged write protected. It can be written only by a privileged access. All bits in this register can be read by both privileged and unprivileged, secure and nonsecure access.

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **NSPRIV**: Privileged protection for nonsecure registers
0: Nonsecure flash registers can be read and written by privileged or unprivileged access.
1: Nonsecure flash registers can be read and written by privileged access only.

Bit 0 **SPRIV**: Privileged protection for secure registers
This bit is secure write protected. It can only be written by a secure privileged access when TrustZone is enabled (TZEN = 1).
0: Secure flash registers can be read and written by privileged or unprivileged access.
1: Secure flash registers can be read and written by privileged access only.

7.9.25 FLASH secure HDP extension register (FLASH_SECHDPEXTR)

Address offset: 0x0C8

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

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

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **HDP_PEXT[6:0]:** Number of secure HDP extension area pages.
This field contains the number of flash pages assigned to the HDP extension area.
When HDPEN is enabled, the HDP extension area starts at HDP_PEND + 1.
When HDPEN is disabled the HDP extension area starts at SECWM_PSTRT.

7.9.26 FLASH privilege block based register x (FLASH_PRIVBBRx)

Address offset: 0x0D0 + 0x4 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

This register is privileged. It can be read written only by a privileged access. Individual register bits can be protected against nonsecure access (refer to *Table 69*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIVBB31	PRIVBB30	PRIVBB29	PRIVBB28	PRIVBB27	PRIVBB26	PRIVBB25	PRIVBB24	PRIVBB23	PRIVBB22	PRIVBB21	PRIVBB20	PRIVBB19	PRIVBB18	PRIVBB17	PRIVBB16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIVBB15	PRIVBB14	PRIVBB13	PRIVBB12	PRIVBB11	PRIVBB10	PRIVBB9	PRIVBB8	PRIVBB7	PRIVBB6	PRIVBB5	PRIVBB4	PRIVBB3	PRIVBB2	PRIVBB1	PRIVBB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIVBB[31:0]:** page privileged/unprivileged attribution (y = 0 to 31)
Each bit is used to set one page privilege attribution in flash.
0: Page (32 x (x - 1) + y) in flash accessible by unprivileged access
1: Page (32 x (x - 1) + y) in flash only accessible by privileged access

7.9.27 FLASH OEM1 key register 1 (FLASH_OEM1KEYR1)

Address offset: 0x110

ST factory programmed value: 0xFFFF FFFF

Reset value: 0x0000 0000

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

This register is nonsecure. It can be written by both secure and nonsecure access. This register is read as zero. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM1KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEM1KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM1KEY[31:0]**: OEM1 key least significant bytes

7.9.28 FLASH OEM1 key register 2 (FLASH_OEM1KEYR2)

Address offset: 0x114

ST factory programmed value: 0xFFFF FFFF

Reset value: 0xXXXX XXXX

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

This register is nonsecure. It can be written by both secure and nonsecure access. This register is read as zero. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM1KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEM1KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM1KEY[63:32]**: OEM1 key middle bytes

7.9.29 FLASH OEM1 key register 3 (FLASH_OEM1KEYR3)

Address offset: 0x118

ST factory programmed value: 0xFFFF FFFF

Reset value: 0xXXXX XXXX

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

This register is nonsecure. It can be written by both secure and nonsecure access. This register is read as zero. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM1KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEM1KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM1KEY[95:64]**: OEM1 key middle bytes

7.9.30 FLASH OEM1 key register 4 (FLASH_OEM1KEYR4)

Address offset: 0x11C

ST factory programmed value: 0xFFFF FFFF

Reset value: 0xXXXX XXXX

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

This register is nonsecure. It can be written by both secure and nonsecure access. This register is read as zero. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM1KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEM1KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM1KEY[127:96]**: OEM1 key most significant bytes

7.9.31 FLASH OEM2 key register 1 (FLASH_OEM2KEYR1)

Address offset: 0x120

ST factory programmed value: 0xFFFF FFFF

Reset value: 0xXXXX XXXX

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

This register is nonsecure. It can be written by both secure and nonsecure access. This register is read as zero. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM2KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEM2KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM2KEY[31:0]**: OEM2 key least significant bytes

7.9.32 FLASH OEM2 key register 2 (FLASH_OEM2KEYR2)

Address offset: 0x124

ST factory programmed value: 0xFFFF FFFF

Reset value: 0xXXXX XXXX

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

This register is nonsecure. It can be written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM2KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEM2KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM2KEY[63:32]**: OEM2 key middle bytes

7.9.33 FLASH OEM2 key register 3 (FLASH_OEM2KEYR3)

Address offset: 0x128

ST factory programmed value: 0xFFFF FFFF

Reset value: 0xXXXX XXXX

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

This register is nonsecure. It can be written by both secure and nonsecure access. This register is read as zero. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM2KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEM2KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM2KEY[95:64]**: OEM2 key middle bytes

7.9.34 FLASH OEM2 key register 4 (FLASH_OEM2KEYR4)

Address offset: 0x12C

ST factory programmed value: 0xFFFF FFFF

Reset value: 0xXXXX XXXX

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

This register is nonsecure. It can be written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM2KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEM2KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM2KEY[127:96]**: OEM2 key most significant bytes

7.9.35 FLASH OEM key status register (FLASH_OEMKEYSR)

Address offset: 0x130

Reset value: 0x00XX 00XX

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

This register is nonsecure. It can be read by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OEM2KEYCRC[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OEM1KEYCRC[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **OEM2KEYCRC[7:0]**: OEM2 key CRC value

CRC is automatically computed by hardware when loading the OEM2 key. CRC computation is an checksum using the standard CRC-8-CCITT algorithm X8 + X2 + X + 1 (according the convention). Source code is available in *Section 7.6.2: Readout protection (RDP)*.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **OEM1KEYCRC[7:0]**: OEM1 key CRC value

CRC is automatically computed by hardware when loading the OEM1 key. CRC computation is an checksum using the standard CRC-8-CCITT algorithm X8 + X2 + X + 1 (according the convention). Source code is available in *Section 7.6.2: Readout protection (RDP)*.

7.9.36 FLASH register map

Table 72. FLASH register map and reset values

Offset	Register	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
0x000	FLASH_ACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEP_PD	Res.	PDREQ	LPM	Res.	Res.	PRFTEN	Res.	Res.	Res.	Res.	LATENCY [3:0]			
	Reset value																		0		0	0			0					0	0	0	1
0x004	Reserved	Reserved																															
0x008	FLASH_NSKEYR	NSKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	FLASH_SECKEYR	SECKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	FLASH_OPTKEYR	OPTKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	Reserved	Reserved																															
0x018	FLASH_PDKEY1R	PDKEY1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	Reserved	Reserved																															
0x020	FLASH_NSSR	Res.	Res.	Res.	Res.	Res.	OPBRK	Res.	Res.	Res.	Res.	Res.	PD	OEM2LOCK	OEM1LOCK	WDW	BSY	Res.	Res.	OPTWERR	Res.	Res.	Res.	Res.	Res.	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res.	OPERR	EOP
	Reset value						0						0	X	X	X	X			0						0	0	0	0	0		0	0
0x024	FLASH_SECSR	Res.	Res.	Res.	Res.	Res.	OPBRK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDW	BSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res.	OPERR	EOP
	Reset value						0									0	0									0	0	0	0	0		0	0
0x028	FLASH_NSCR1	LOCK	OPTLOCK	Res.	Res.	OBL_LAUNCH	Res.	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	Res.	OPTSTRT	STRT	Res.	BWR	Res.	Res.	Res.	Res.	PNB[6:0]						MER	PER	PG
	Reset value	1	1			0		0	0							0	0		0					0	0	0	0	0	0	0	0	0	0
0x02C	FLASH_SECCR1	LOCK	Res.	INV	Res.	Res.	Res.	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT	Res.	BWR	Res.	Res.	Res.	Res.	PNB[6:0]						MER	PER	PG
	Reset value	1		0				0	0								0		0					0	0	0	0	0	0	0	0	0	0
0x030	FLASH_ECCR	ECCD	ECCC	Res.	Res.	Res.	Res.	Res.	ECCCIE	Res.	SYSF_ECC	Res.	Res.	ADDR_ECC[19:0]																			
	Reset value	0	0						0		0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	FLASH_OPSR	CODE_OP[2:0]			ET_OP[5:0]						SYSF_OP	Res.	Res.	ADDR_OP[19:0]																			
	Reset value	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 72. FLASH register map and reset values (continued)

Offset	Register	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
0x038	FLASH_NSCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPA	Res.	Res.	Res.	Res.	Res.	Res.	ES	PS
	Reset value																								0							0	0
0x03C	FLASH_SECCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPA	Res.	Res.	Res.	Res.	Res.	Res.	ES	PS
	Reset value																								0							0	0
0x040	FLASH_OPTR	TZEN	Res.	Res.	Res.	NBOOT0	NSWBOOT0	SRAM2_RST	SRAM2_PE	Res.	Res.	Res.	Res.	Res.	IWDG_STDBY	IWDG_STOP	IWDG_SW	SRAM1_RST	Res.	NRST_STDBY	NRST_STOP	Res.	BOR_LEV[2:0]		RDP[7:0]								
	Reset value	X				X	X	X	X						X	X	X	X		X	X		X	X	X	X	X	X	X	X	X	X	X
0x044	FLASH_ NSBOOTADD0R	NSBOOTADD0[24:0]																									Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X							
0x048	FLASH_ NSBOOTADD1R	NSBOOTADD1[24:0]																									Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X							
0x04C	FLASH_ SECBOOTADD0R	SECBOOTADD0[24:0]																									Res.	Res.	Res.	Res.	Res.	Res.	BOOT_LOCK
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X							X
0x050	FLASH_ SECWMR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM_PSTRT[6:0]						
	Reset value										X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x054	FLASH_ SECWMR2	HDPEN[7:0]								Res.	HDP_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X																
0x058	FLASH_WRPAR	UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPA_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPA_PSTRT[6:0]						
	Reset value	X									X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x05C	FLASH_WRPBR	UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPB_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPB_PSTRT[6:0]						
	Reset value	X									X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x070 to 0x07C	Reserved	Reserved																															
0x080	FLASH_ SECBBR1	SECBB31	SECBB30	SECBB29	SECBB28	SECBB27	SECBB26	SECBB25	SECBB24	SECBB23	SECBB22	SECBB21	SECBB20	SECBB19	SECBB18	SECBB17	SECBB16	SECBB15	SECBB14	SECBB13	SECBB12	SECBB11	SECBB10	SECBB9	SECBB8	SECBB7	SECBB6	SECBB5	SECBB4	SECBB3	SECBB2	SECBB1	SECBB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	FLASH_ SECBBR2	SECBB31	SECBB30	SECBB29	SECBB28	SECBB27	SECBB26	SECBB25	SECBB24	SECBB23	SECBB22	SECBB21	SECBB20	SECBB19	SECBB18	SECBB17	SECBB16	SECBB15	SECBB14	SECBB13	SECBB12	SECBB11	SECBB10	SECBB9	SECBB8	SECBB7	SECBB6	SECBB5	SECBB4	SECBB3	SECBB2	SECBB1	SECBB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 72. FLASH register map and reset values (continued)

Offset	Register	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
0x088	FLASH_ SECBBR3	SECBB31	SECBB30	SECBB29	SECBB28	SECBB27	SECBB26	SECBB25	SECBB24	SECBB23	SECBB22	SECBB21	SECBB20	SECBB19	SECBB18	SECBB17	SECBB16	SECBB15	SECBB14	SECBB13	SECBB12	SECBB11	SECBB10	SECBB9	SECBB8	SECBB7	SECBB6	SECBB5	SECBB4	SECBB3	SECBB2	SECBB1	SECBB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	FLASH_ SECBBR4	SECBB31	SECBB30	SECBB29	SECBB28	SECBB27	SECBB26	SECBB25	SECBB24	SECBB23	SECBB22	SECBB21	SECBB20	SECBB19	SECBB18	SECBB17	SECBB16	SECBB15	SECBB14	SECBB13	SECBB12	SECBB11	SECBB10	SECBB9	SECBB8	SECBB7	SECBB6	SECBB5	SECBB4	SECBB3	SECBB2	SECBB1	SECBB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090 to 0x0BC	Reserved	Reserved																															
0x0C0	FLASH_ SECHDPCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPEXT_ACCDIS[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP_ACCDIS[7:0]							
	Reset value									1	0	1	0	0	0	1	1									1	0	1	0	0	0	1	1
0x0C4	FLASH_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NSPRIV	SPRIV
	Reset value																															0	0
0x0C8	FLASH_ SECHDPEXTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP_PEXT[6:0]						
	Reset value																										0	0	0	0	0	0	0
0x0CC	Reserved	Reserved																															
0x0D0	FLASH_ PRIVBBR1	PRIVBB31	PRIVBB30	PRIVBB29	PRIVBB28	PRIVBB27	PRIVBB26	PRIVBB25	PRIVBB24	PRIVBB23	PRIVBB22	PRIVBB21	PRIVBB20	PRIVBB19	PRIVBB18	PRIVBB17	PRIVBB16	PRIVBB15	PRIVBB14	PRIVBB13	PRIVBB12	PRIVBB11	PRIVBB10	PRIVBB9	PRIVBB8	PRIVBB7	PRIVBB6	PRIVBB5	PRIVBB4	PRIVBB3	PRIVBB2	PRIVBB1	PRIVBB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D4	FLASH_ PRIVBBR2	PRIVBB31	PRIVBB30	PRIVBB29	PRIVBB28	PRIVBB27	PRIVBB26	PRIVBB25	PRIVBB24	PRIVBB23	PRIVBB22	PRIVBB21	PRIVBB20	PRIVBB19	PRIVBB18	PRIVBB17	PRIVBB16	PRIVBB15	PRIVBB14	PRIVBB13	PRIVBB12	PRIVBB11	PRIVBB10	PRIVBB9	PRIVBB8	PRIVBB7	PRIVBB6	PRIVBB5	PRIVBB4	PRIVBB3	PRIVBB2	PRIVBB1	PRIVBB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D8	FLASH_ PRIVBBR3	PRIVBB31	PRIVBB30	PRIVBB29	PRIVBB28	PRIVBB27	PRIVBB26	PRIVBB25	PRIVBB24	PRIVBB23	PRIVBB22	PRIVBB21	PRIVBB20	PRIVBB19	PRIVBB18	PRIVBB17	PRIVBB16	PRIVBB15	PRIVBB14	PRIVBB13	PRIVBB12	PRIVBB11	PRIVBB10	PRIVBB9	PRIVBB8	PRIVBB7	PRIVBB6	PRIVBB5	PRIVBB4	PRIVBB3	PRIVBB2	PRIVBB1	PRIVBB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0DC	FLASH_ PRIVBBR4	PRIVBB31	PRIVBB30	PRIVBB29	PRIVBB28	PRIVBB27	PRIVBB26	PRIVBB25	PRIVBB24	PRIVBB23	PRIVBB22	PRIVBB21	PRIVBB20	PRIVBB19	PRIVBB18	PRIVBB17	PRIVBB16	PRIVBB15	PRIVBB14	PRIVBB13	PRIVBB12	PRIVBB11	PRIVBB10	PRIVBB9	PRIVBB8	PRIVBB7	PRIVBB6	PRIVBB5	PRIVBB4	PRIVBB3	PRIVBB2	PRIVBB1	PRIVBB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0E0 to 0x10C	Reserved	Reserved																															
0x110	FLASH_ OEM1KEYR1	OEM1KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x114	FLASH_ OEM1KEYR2	OEM1KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 72. FLASH register map and reset values (continued)

Offset	Register	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
0x118	FLASH_ OEM1KEYR3	OEM1KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11C	FLASH_ OEM1KEYR4	OEM1KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x120	FLASH_ OEM2KEYR1	OEM2KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x124	FLASH_ OEM2KEYR2	OEM2KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x128	FLASH_ OEM2KEYR3	OEM2KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x12C	FLASH_ OEM2KEYR4	OEM2KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x130	FLASH_OEMKEYS R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OEM2KEYCRC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OEM1KEYCRC[7:0]							
	Reset value									X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X
0x134 to 0x3FC	Reserved	Reserved																															

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

8 Instruction cache (ICACHE)

8.1 ICACHE introduction

The instruction cache (ICACHE) is implemented on the C-AHB code bus of the Cortex®-M33 processor to improve performance when fetching instructions and data from internal and external memories.

Some specific features, like dual master ports, hit-under-miss, and critical-word-first refill policy, result in close to zero-wait-state performance in most use cases.

8.2 ICACHE main features

The main features of ICACHE are listed below:

- Bus interface
  - One 32-bit AHB slave port, the execution port (input from Cortex®-M33 C-AHB code interface)
  - Two AHB master ports: master1 and master2 ports, with outputs to Fast and Slow buses of main AHB bus matrix, respectively
  - One 32-bit AHB slave port for control (input from AHB peripherals interconnect, for ICACHE registers access)
- Cache access
  - 0 wait-state on hits
  - Hit-under-miss capability: ability to serve processor requests (access to cached data) during an ongoing line refill due to a previous cache miss
  - Dual master access: feature used to decouple the traffic according to targeted memory. For example, the ICACHE assigns fast traffic (addressing flash memories and SRAMs) to the AHB master1 port, and slow traffic (addressing external memories) to the AHB master2 port, thus preventing processor stalls on line refills from external memories. This allows ISR (interrupt service routine) fetching on internal flash memory to take place in parallel with a cache line refill from external memories.
  - Minimal impact on interrupt latency, thanks to dual master
  - Optimal cache line refill thanks to WRAPw bursts of the size of the cache line (32-bit word size, **w**, aligned on cache line size)
  - n-way set-associative default configuration with possibility to configure as 1-way, means direct mapped cache, for applications needing very-low-power consumption profile
- Memory address remap
  - Possibility to remap input address falling into up to four memory regions (used to remap aliased code in external memories to the code region, for execution from the C-AHB code interface)
- Replacement and refill
  - pLRU-t (pseudo-least-recently-used, based on binary tree) replacement policy, algorithm with best complexity/performance balance
  - Critical-word-first refill policy, minimizing processor stalls

  - Possibility to configure burst type of AHB memory transaction for remapped regions: INCRw or WRAPw (size w aligned on cache line size)
- Performance counters

  The ICACHE implements two performance counters:
  - Hit monitor counter (32-bit)
  - Miss monitor counter (16-bit)
- Error management
  - Possibility to detect an unexpected cacheable write access, to flag an error, and optionally, to raise an interrupt
- TrustZone® security support
- Maintenance operation
  - Cache invalidate: full cache invalidation, fast command, noninterruptible

8.3 ICACHE implementation

Table 73. ICACHE features

Feature	ICACHE1
Number of ways	2
Cache size	4 Kbytes
Cache line width	16 bytes
Range granularity of memory regions to be remapped	2 Mbytes
Number of regions to remap	4
Data size of AHB slave interface	32 bits
Data size of AHB fast master1 interface	32 bits
Data size of AHB slow master2 interface	32 bits

8.4 ICACHE functional description

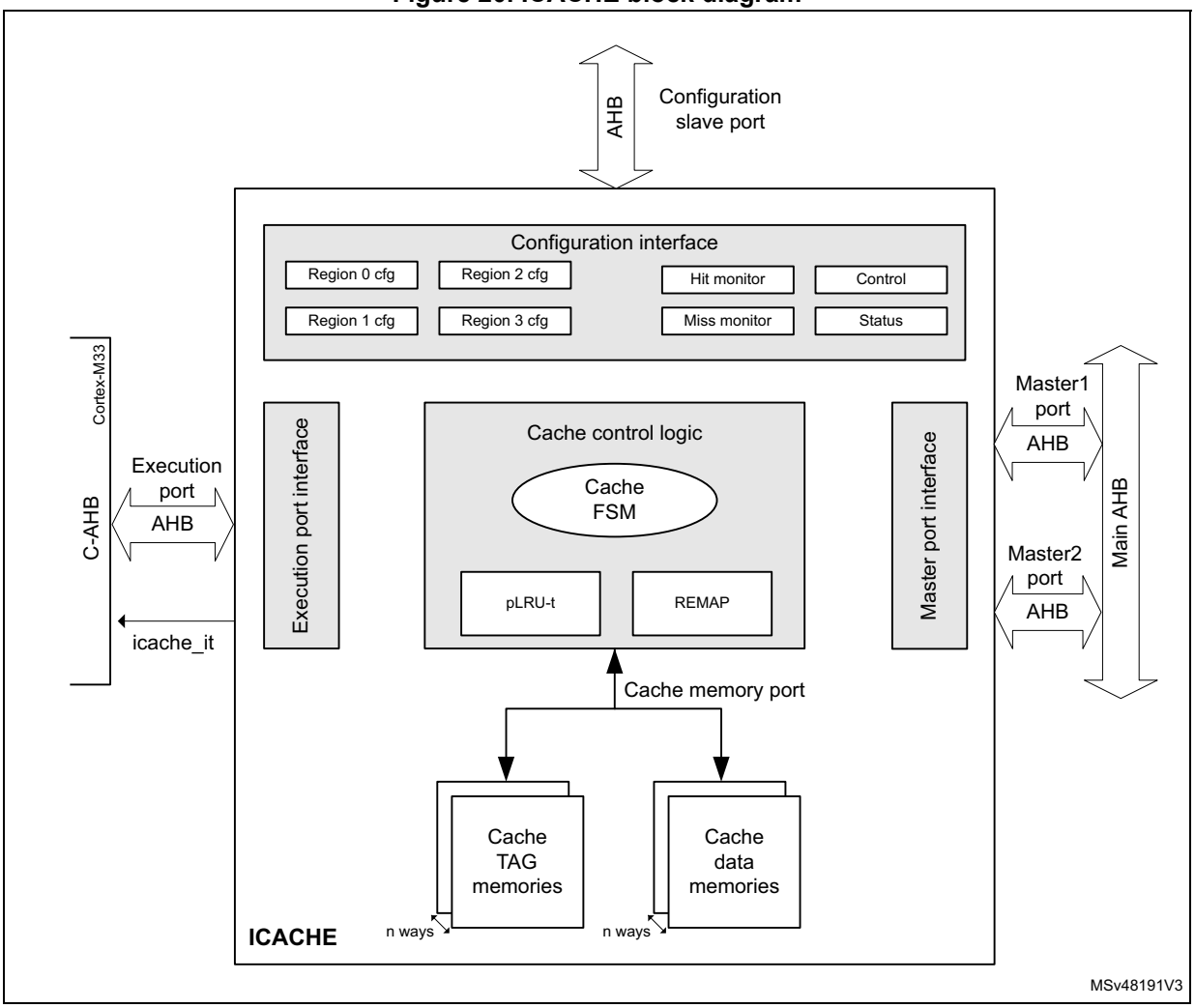
The purpose of this block is to cache instruction fetches or instruction memories loads, coming from the processor. As such, the ICACHE manages only cacheable read transactions, and does not manage cacheable write transactions.

The noncacheable transactions (both read and write) bypass the ICACHE.

For the error management purpose, if a write cacheable transaction is presented (this happens only in case of bad software programming), the ICACHE sets an error flag and, if enabled, raises an interrupt to the processor.

8.4.1 ICACHE block diagram

Figure 20. ICACHE block diagram



8.4.2 ICACHE reset and clocks

The ICACHE is clocked on the Cortex®-M33 C-AHB bus clock.

When the ICACHE reset signal is released, a cache invalidate procedure is automatically launched, making the ICACHE busy (ICACHE_SR = 0x0000 0001).

When this procedure is finished:

- the ICACHE is invalidated: “cold cache”, with all cache line valid bits = 0 (ICACHE must be filled up)
- ICACHE_SR = 0x0000 0002 (reflecting the cache is no longer busy)
- the ICACHE is disabled: the EN bit in ICACHE_CR holds its reset state (= 0).

Note: When disabled, the ICACHE is bypassed, except the remapping mechanism that is still functional: slave input requests (remapped or not) are forwarded to the master port(s).

8.4.3 ICACHE TAG memory

The ICACHE TAG memory contains:

- address tags indicating which data are contained in the cache data memories
- validity bits

There is one valid bit per cache line (per way).

The valid bit is set when a cache line is refilled (after a miss).

Valid bits are reset in any of the below cases:

- after the ICACHE reset is released
- when the cache is disabled, by setting EN = 0 in ICACHE_CR (by software)
- when executing an ICACHE invalidate command, by setting CACHEINV = 1 in ICACHE_CR (by software)

When a cacheable transaction is received at the execution input port, its AHB address (HADDR_in) is split into the following fields (see *Table 74* for B and W definitions):

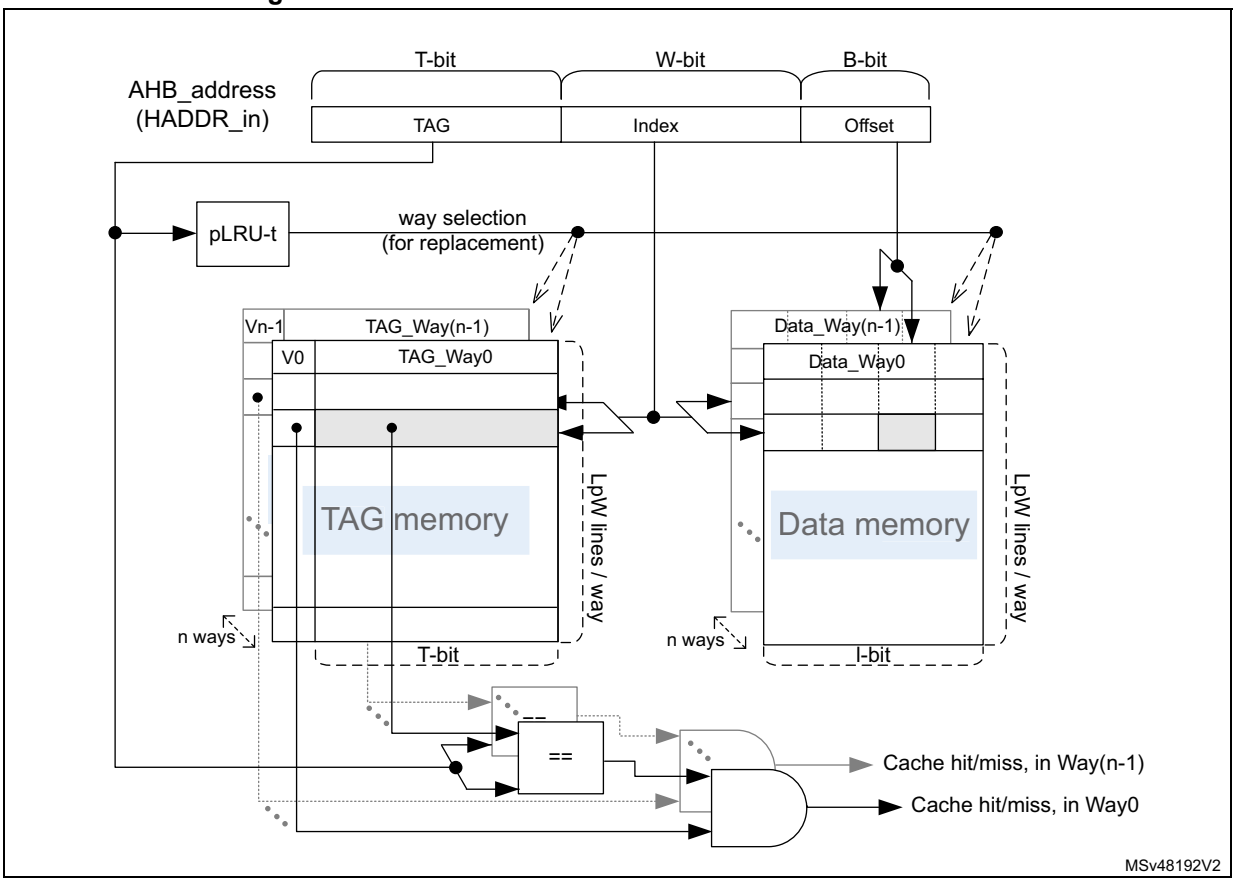
- HADDR_in[B-1:0]: address byte offset, indicates which byte to select inside a cache line.
- HADDR_in[B+W-1:B]: address way index, indicates which cache line to select inside each way.
- HADDR_in[31:B+W]: tag address, to be compared to the TAG memory address to check if the requested data is already available (meaning valid) inside the ICACHE.

The following table gives a summary of the ICACHE main parameters for TAG memory dimensioning. *Figure 21* shows the functional view of TAG and data memories, for an n-way set associative ICACHE.

Table 74. TAG memory dimensioning parameters for n-way set associative operating mode (default)

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	8 Kbytes = 8192 bytes
Cache number of ways	n	2
Cache line size	L-byte = l-bit (l = 8 x L)	16-byte = 128-bit
Number of cache lines	LpW = s / (n x L) lines / way	256 lines / way
Address byte offset size	B = $\log_2$(L) bit	4-bit
Address way index size	W = $\log_2$(LpW) bit	8-bit
TAG address size	T = (32 - W - B) bit	20-bit

Figure 21. ICACHE TAG and data memories functional view



8.4.4 Direct-mapped ICACHE (1-way cache)

The default configuration (at reset) is an n-way set associative cache (WAYSEL = 1 in ICACHE_CR), but the user can configure the ICACHE as direct mapped by writing WAYSEL = 0 (only possible when the cache is disabled, EN = 0 in ICACHE_CR).

The following table gives a summary of ICACHE main parameters for TAG memory when the direct-mapped cache operating mode is selected.

Table 75. TAG memory dimensioning parameters for direct-mapped cache mode

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	8 Kbytes = 8192 bytes
Cache number of ways	1	1
Cache line size	L-byte = l-bit (l = 8 x L)	16-byte = 128-bit
Number of cache lines	LpW = s / L lines	512 lines
Address byte offset size	B = $\log_2$(L) bit	4-bit
Address way index size	W = $\log_2$(LpW) bit	9-bit
TAG address size	T = (32 - W - B) bit	19-bit

All cache operations (such as read, refill, remapping, invalidation) remain the same in the direct-mapped configuration. The only difference is the absence of a replacement algorithm in case of line eviction (as explained in *Section 8.4.8*): only one way (the unique one) is possible for any data refill.

8.4.5 ICACHE enable

To activate the ICACHE, the EN bit in ICACHE_CR must be set to 1.

When the ICACHE is disabled, it is bypassed and all transactions are copied from the slave to the master ports in the same clock cycle.

It is recommended to initialize or modify the main memory content (region to be later cached) with the ICACHE disabled, and to enable the ICACHE only when this region remains unchanged (an enabled ICACHE detects cacheable write transactions as errors).

To ensure performance determinism, it is recommended to wait for the end of a potential cache invalidate procedure before enabling the ICACHE. This procedure occurs when the hardware reset signal is released, when CACHEINV is set, or when EN is cleared in ICACHE_CR. During the procedure, BUSYF is set in ICACHE_SR, and once finished, BUSYF is cleared and BSYENDF is set in the same register (raising the ICACHE interrupt if enabled on such a busy end condition).

The software must test BUSYF and/or BSYENDF values before enabling the ICACHE. Else, if the ICACHE is enabled before the end of an invalidate procedure, any cache access (while BUSYF = 1) is treated as noncacheable, and its performance depends on the main memory access time.

The address remapping is performed, whether the ICACHE is enabled or not, if the input transaction address falls into the memory regions defined and enabled in ICACHE_CRRx (see *Figure 22*).

The ICACHE is, by default, disabled at boot.

8.4.6 Cacheable and noncacheable traffic

The ICACHE is developed for the Cortex®-M33 core. It is placed on the C-AHB bus, and thus caches the code memory region, ranging from address 0x0000 0000 to 0x1FFF FFFF of the memory map.

To make other memory regions cacheable, the ICACHE supports a memory region remapping feature, used to define up to four external memory regions, whose addresses have an alias in the code region. Addressing these memory regions through their code alias address allows the memory request to be routed to the C-AHB bus, and to be managed by the ICACHE.

Any external memory space physically mapped at an address in the range [0x6000 0000:0xAFFF FFFF] can be aliased with an address in the range [0x0000 0000:0x07FF FFFF] or [0x1000 0000:0x1FFF FFFF].

For a given memory request in the code region, the ICACHE implements the address remapping functionality first. If aliased, it is the remapped address, which is then cached, and, if needed, provided to the master port to address the main AHB bus matrix. The destination physical address does not need further manipulation on the AHB bus.

The remapping functionality is available also for noncacheable traffic, and when the cache is disabled.

Additional details on address remapping are provided in *Section 8.4.7*.

An incoming memory request to the ICACHE is defined as cacheable according to its AHB transaction memory lookup attribute, as shown in *Table 76*. This AHB attribute depends on the MPU (memory protection unit) programming for the addressed region.

Table 76. ICACHE cacheability for AHB transaction

AHB lookup attribute	Cacheability
1	Cacheable
0	Noncacheable

In the case of a noncacheable access (read or write), the ICACHE is bypassed. The AHB transaction is propagated unchanged to the master output port, except the transaction address, which may be modified due to the address remapping feature (see *Section 8.4.7*).

The bypass, and eventual remap logic, does not increase the latency of the access to the targeted memory.

In the case of a cacheable access, the ICACHE behaves as explained in *Section 8.4.8*.

Cacheable memory regions are defined and programmed by the user in the MPU responsible for the generation of the AHB attribute signals for any transaction addressing a given region.

Table 77 summarizes programmable configurations of various memories.

Table 77. Memory configurations

Memory	Cacheable (MPU programming)	Remapped in the ICACHE (ICACHE_CRRx programming)
Flash	Yes or no	Not required
SRAM	Not recommended	Not required
External memories	Yes	Required
	No	Required if the user wants code in external memories fetched on C-AHB bus (else on S-AHB bus)

8.4.7 Address remapping

The ICACHE allows an alias address to be defined in the code region for up to four external memory regions.

The address remapping is applied on the code alias address, transforming it into the destination external physical address.

The remapping operation is fully software configurable by programming ICACHE_CRRx (x = 0 to 3, number of remapped regions). This programming can be done only when the ICACHE is disabled.

Each region x can be individually enabled with REN in ICACHE_CRRx. Once enabled, the remap operation occurs even if the ICACHE is disabled, or if the transaction is noncacheable.

Remap regions can have different size: each can be programmed in RSIZE of its ICACHE_CRRx. The size of each region is a power of two multiple of range granularity (2 Mbytes), with a minimum of 2 Mbytes, and a maximum of 128 Mbytes.

The address remapping mechanism is based on the matching of an incoming AHB address (HADDR_in) with a given code subregion base-address, and the modification of this address into its (remapped) physical address, as follows:

- HADDR_in belongs to region x if HADDR_in[31:RI] = 000:BASEADDR[28:RI], where:
  - 000:BASEADDR is the code subregion base-address programmed in BASEADDR of ICACHE_CRRx.
  - RI defines the number of significant bits to consider. RI = $\log_2$ (region size) with a minimum value of 21 (for a 2-Mbyte region) and a maximum value of 27 (for a 128-Mbyte region).
- If the region x is enabled, the master port output AHB address (HADDR_out) is composed by concatenating the two below parts:
  - REMAPADDR[31:RI] field of ICACHE_CRRx as MSBs
  - HADDR_in[RI-1:0] as LSBs

Figure 22 describes the matching and the output address generation.

Figure 22. ICACHE remapping address mechanism

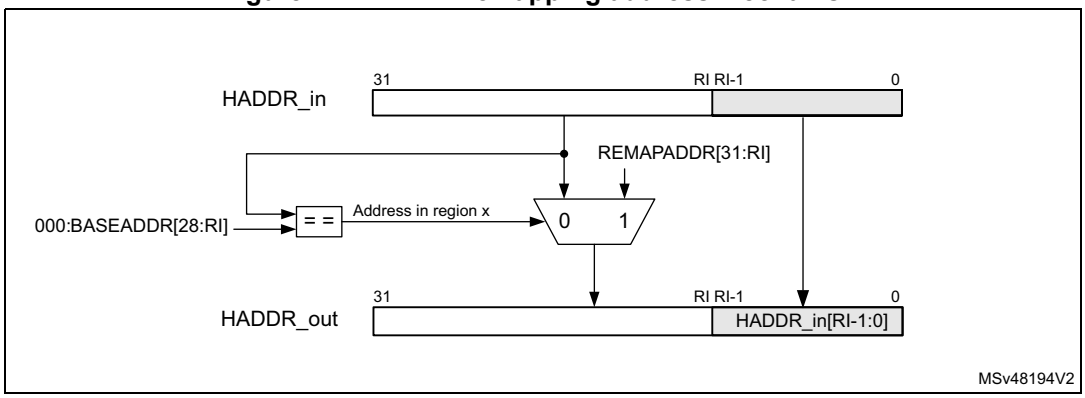


Table 78 summarizes all possible configurations of BASEADDR and REMAPADDR sizes (number of significant MSBs) in ICACHE_CRRx, depending upon RSIZE.

Table 78. ICACHE remap region size, base address, and remap address

Region size (Mbytes)	Base address size (MSBs)	Remap address (MSBs)
2	8	11
4	7	10
8	6	9
16	5	8
32	4	7
64	3	6
128	2	5

Care must be taken while programming BASEADDR and REMAPADDR in ICACHE_CRRx: if the programmed value is bigger than expected (number of MSBs, see *Table 78*), the unneeded extra LSBs are ignored.

Typical remapping example: a 128-Mbyte external RAM region physically located in the external address range [0x9000 0000:0x97FF FFFF], remapped in the code section range [0x1800 0000:0x1FFF FFFF]:

- REMAPADDR[31:21] = 0x480
- BASEADDR[28:21] = 0xC0
- HADDR_in[31:27] is compared to 000:BASEADDR[28:27], while HADDR_in and BASEADDR[26:21] are ignored.

If the comparison matches:

- HADDR_out[31:27] gets REMAPADDR[31:27] (in place of HADDR_in[31:27])
- HADDR_out[26:0] gets HADDR_in[26:0]

The software can program the kind of AHB burst generated by the ICACHE master ports on the bus matrix (for cache line refill), by setting HBURST in ICACHE_CRRx with:

- WRAP for remapped external memories accessed through interfaces that can support WRAP burst mode, providing the benefit of the critical-word-first feature performance:
  - WRAP burst size = cache line size
  - WRAP burst start address = word address of the first data requested by the core
- INCR: INCR burst mode for external memories accessed through the interfaces that do not support WRAP burst mode (losing the benefit of critical-word-first feature):
  - INCR burst size = cache line size
  - INCR burst start address = address aligned on the boundary of the cache line containing the requested word.

Note: Coherency is needed when programming the SAU (secure attribution unit) and the MPU (memory protection unit) attributes for both the external regions and their aliased code subregions.

8.4.8 Cacheable accesses

When the ICACHE receives a cacheable transaction from the Cortex®-M33, it checks if the address requested is present in its TAG memory, and if the corresponding cache line is valid.

There are then three alternatives:

- The address is present inside the TAG memory, the cache line is valid: **cache hit**, the data is read from the cache and provided to the processor in the same cycle.
- The address is not present in the TAG memory: **cache miss**, the data is read from the main memory and provided to the processor, and a cache line refill is performed.

  The critical-word-first policy ensures minimum wait cycles for the processor, since read data can be provided while the cache still performs a cache line refill (associated latency is the latency of fetching one word from the main memory).

  The burst generated on the ICACHE master bus is WRAPw (w being the cache line width, in words) if no address remap occurs. If an address remap occurs, the kind of burst depends upon the HBURST programmed in corresponding ICACHE_CRRx.

  The AHB transaction attributes are also propagated to the main AHB bus matrix on the master port selected for the line refill.

- The address is not present in TAG memory, but belongs to the refill burst from the main memory currently ongoing: **cache hit** (hit-under-miss feature).

  This happens during cache-line refill. The ICACHE can provide the requested data as soon as data are available at its master interface, thus avoiding a miss (fetching data from the main memory).

In the case of cache refill (due to cache miss), the ICACHE selects which cache line is written with the refill data:

- In direct map (1-way) mode, only one line can be used to store the refill data: the line pointed by the index of the input address.
- In n-way set associative mode, one line among n can be used (the line pointed by the address index, in each of the n ways). The way selection is based on a pLRU-t replacement algorithm that points, for each index, on the way candidate for the next refill.

If the cache line where the refill data must be written is already valid, the targeted cache line must be invalidated first. This is true whatever the direct map or n-way set associative cache mode.

8.4.9 Dual-master cache

The ICACHE implements a dual-port AHB master on the main AHB bus matrix: master1 and master2 ports. This is used to split the traffic going to different destination memories.

The nonremapped traffic goes systematically to master1 port. The remapped traffic can be routed on the master2 port by programming MSTSEL in ICACHE_CRRx (on a region basis).

The code can typically be fetched as follows:

- internal flash memory and internal SRAM on master1 port (Fast bus)
- external flash memory/RAM on master2 port (Slow bus)

For systems not implementing external memories, the traffic to the internal flash memory can be decoupled from the traffic to the internal SRAM (when remapped by the ICACHE). This feature is used to prevent further processor stalls on misses.

Alongside with hit-under-miss, this dual-master feature allows the processor to have an alternative path in case of fetching from different memories.

8.4.10 ICACHE security

The ICACHE implements an Armv8-M TrustZone.

ICACHE configuration registers are protected at system level.

8.4.11 ICACHE maintenance

The software can invalidate the whole content of the ICACHE by programming CACHEINV in the ICACHE_CR register.

When CACHEINV = 1, the ICACHE control logic sets the BUSYF flag in ICACHE_SR and launches the invalidate cache operation, resetting each TAG valid bit to 0 (one valid bit per cache line). CACHEINV is automatically cleared.

Once the invalidate operation is finished (all valid bits reset to 0), the ICACHE automatically clears BUSYF, and sets BSYENDF in the ICACHE_SR register.

If enabled on this flag condition (BSYENDIE = 1 in ICACHE_IER), the ICACHE interrupt is raised. Then, the (empty) cache is available again.

8.4.12 ICACHE performance monitoring

The ICACHE provides the following monitors for performance analysis:

- The 32-bit hit monitor counts the cacheable AHB-transactions on the slave cache port that hits the ICACHE content.

  It also takes into account all accesses whose address is present in the TAG memory or in the refill buffer (due to a previous miss, and whose data is coming, or is soon to come, from the cache master port) (see *Section 8.4.8*).
- The 16-bit miss monitor counts the cacheable AHB-transactions on the slave cache port that misses the ICACHE content.

  It also takes into account all accesses whose address is not present neither in the TAG memory nor in the refill buffer.

Upon reaching their maximum values, these monitors do not wrap over.

Hit and miss monitors can be enabled and reset by software allowing the analysis of specific pieces of code.

The software can perform the following tasks:

- Enable/stop the hit monitor through HITMEN in ICACHE_CR.
- Reset the hit monitor by setting HITMRST in ICACHE_CR.
- Enable/stop the miss monitor through MISSMEN in ICACHE_CR.
- Reset the miss monitor by setting MISSMRST in ICACHE_CR.

To reduce power consumption, these monitors are disabled (stopped) by default.

8.4.13 ICACHE boot

The ICACHE is disabled (EN = 0 in ICACHE_CR) at boot.

The code remapping at boot is not needed for a Cortex®-M33 as it implements the VTOR (vector tables) that allows a boot start address definition different than 0x0.

Once the boot is finished, the ICACHE can be enabled (software setting EN = 1 in CACHE_CR).

8.5 ICACHE low-power modes

At device level, using the ICACHE reduces the power consumption by fetching instructions from the internal ICACHE most of the time, rather than from the bigger and then more power consuming main memories. This reduction is even higher if the cached main memories are external.

Applications with lower performance profile (in terms of hit ratio) and stringent power consumption constraints may benefit from the lower power consumption of an ICACHE configured as direct mapped. This single-way cache configuration is obtained by programming WAYSEL = 0 in ICACHE_CR (see *Figure 21*). The power consumption is reduced by accessing, for each request, only the necessary cut of TAG and data memories. The cache effect still improves fetch performance, even if for most code execution, it is less efficient than with an n-way set associative cache mode.

8.6 ICACHE error management and interrupts

If an unsupported cacheable write request is detected (functional error), the ICACHE generates an error by setting the ERRF flag in ICACHE_SR. An interrupt is generated if the corresponding interrupt enable bit is set (ERRIE = 1 in ICACHE_IER).

The other possible interrupt generation is at the end of a cache invalidation operation. When the cache-busy state is finished, the ICACHE sets the BSYENDF flag in ICACHE_SR. An interrupt is generated if the corresponding interrupt enable bit is set (BSYENDIE = 1 in ICACHE_IER).

All ICACHE interrupt sources raise the same and unique interrupt signal, icache_it, and then use the same interrupt vector.

Table 79. ICACHE interrupts

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method
ICACHE	Functional error	ERRF in ICACHE_SR	ERRIE in ICACHE_IER	Set CERRF to 1 in ICACHE_FCR
	End of busy state (invalidate finished)	BSYENDF in ICACHE_SR	BSYENDIE in ICACHE_IER	Set CBSYENDF to 1 in ICACHE_FCR

The ICACHE also propagates all AHB bus errors (such as security issues, address decoding issues) from the master1 or master2 port back to the execution port.

8.7 ICACHE registers

8.7.1 ICACHE control register (ICACHE_CR)

Address offset: 0x000

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISS MRST	HITM RST	MISS MEN	HITM EN
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAY SEL	CACHE INV	EN
													rw	w	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **MISSMRST**: miss monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache miss monitor

Bit 18 **HITMRST**: hit monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache hit monitor

Bit 17 **MISSMEN**: miss monitor enable
0: cache miss monitor switched off. Stopping the monitor does not reset it.
1: cache miss monitor enabled

Bit 16 **HITMEN**: hit monitor enable
0: cache hit monitor switched off. Stopping the monitor does not reset it.
1: cache hit monitor enabled

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **WAYSEL**: cache associativity mode selection
This bit allows user to choose ICACHE set-associativity. It can be written by software only when cache is disabled (EN = 0).
0: direct mapped cache (1-way cache)
1: n-way set associative cache (reset value)

Bit 1 **CACHEINV**: cache invalidation
Set by software and cleared by hardware when the BUSYF flag is set (during cache maintenance operation). Writing 0 has no effect.
0: no effect
1: invalidate entire cache (all cache lines valid bit = 0)

Bit 0 **EN**: enable
0: cache disabled
1: cache enabled

8.7.2 ICACHE status register (ICACHE_SR)

Address offset: 0x004

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRF	BSYENDF	BUSYF
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **ERRF**: cache error flag
0: no error
1: an error occurred during the operation (cacheable write)

Bit 1 **BSYENDF**: busy end flag
0: cache busy
1: full invalidate CACHEINV operation finished

Bit 0 **BUSYF**: busy flag
0: cache not busy on a CACHEINV operation
1: cache executing a full invalidate CACHEINV operation

8.7.3 ICACHE interrupt enable register (ICACHE_IER)

Address offset: 0x008

Reset value: 0x0000 0000

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

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **ERRIE**: interrupt enable on cache error

Set by software to enable an interrupt generation in case of cache functional error (cacheable write access)

0: interrupt disabled on error

1: interrupt enabled on error

Bit 1 **BSYENDIE**: interrupt enable on busy end

Set by software to enable an interrupt generation at the end of a cache invalidate operation.

0: interrupt disabled on busy end

1: interrupt enabled on busy end

Bit 0 Reserved, must be kept at reset value.

8.7.4 ICACHE flag clear register (ICACHE_FCR)

Address offset: 0x00C

Reset value: 0x0000 0000

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

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CERRF**: clear cache error flag

Set by software.

0: no effect

1: clears ERRF flag in ICACHE_SR

Bit 1 **CBSYENDF**: clear busy end flag

Set by software.

0: no effect

1: clears BSYENDF flag in ICACHE_SR.

Bit 0 Reserved, must be kept at reset value.

8.7.5 ICACHE hit monitor register (ICACHE_HMONR)

Address offset: 0x010

Reset value: 0x0000 0000

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

Bits 31:0 **HITMON[31:0]**: cache hit monitor counter

8.7.6 ICACHE miss monitor register (ICACHE_MMONR)

Address offset: 0x014

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MISSMON[15:0]**: cache miss monitor counter

8.7.7 ICACHE region x configuration register (ICACHE_CRRx)

Address offset: 0x020 + 0x4 * x, (x = 0 to 3)

Reset value: 0x0000 0200

Define an alias address in the code region for other regions, making them cacheable.

BASEDADDR and REMAPADDR fields are write locked (read only) when EN = 1 in ICACHE_CR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HBURST	Res.	Res.	MSTSEL	Res.	REMAPADDR[31:21]										
rw			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REN	Res.	Res.	Res.	RSIZE[2:0]			Res.	BASEADDR[28:21]							
rw				rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **HBURST**: output burst type for region x
0: WRAP
1: INCR

Bits 30:29 Reserved, must be kept at reset value.

Bit 28 **MSTSEL**: AHB cache master selection for region x
0: no action (master1 selected by default)
1: master2 selected

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **REMAPADDR[31:21]**: remapped address for region x
This field replaces the alias address defined by BASEADDR field.
The only useful bits are [31:RI], where 21 ≤ RI ≤ 27 is the number of bits of RSIZE (see *Section 8.4.7*). If the programmed value has more LSBs, the useless bits are ignored.

Bit 15 **REN**: enable for region x
0: disabled
1: enabled

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:9 **RSIZE[2:0]**: size for region x
000: reserved
001: 2 Mbytes
010: 4 Mbytes
011: 8 Mbytes
100: 16 Mbytes
101: 32 Mbytes
110: 64 Mbytes
111: 128 Mbytes

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **BASEADDR[28:21]**: base address for region x
This alias address is replaced by REMAPADDR field.
The only useful bits are [28:RI], where 21 ≤ RI ≤ 27 is the number of bits of RSIZE (see *Section 8.4.7*). If the programmed value has more LSBs, the useless bits are ignored.

8.7.8 ICACHE register map

Table 80. ICACHE register map and reset values

Offset	Register name	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
0x000	ICACHE_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMRST	HITMRST	MISSMEN	HITMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAYSEL	CACHEINV	EN
	Reset value													0	0	0	0														1	0	0
0x004	ICACHE_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRF	BSYENDF	BUSYF
	Reset value																														0	0	1

Table 80. ICACHE register map and reset values (continued)

Offset	Register name	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
0x008	ICACHE_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRIE	BSYENDIE	Res.
	Reset value																														0	0	
0x00C	ICACHE_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERRF	CBSYENDF	Res.
	Reset value																														0	0	
0x010	ICACHE_HMONR	HITMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	ICACHE_MMONR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMON[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018-0x01C	Reserved	Reserved																															
0x020	ICACHE_CRR0	HBURST	Res.	Res.	MSTSEL	Res.	REMAPADDR[31:21]											REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	BASEADDR[28:21]							
	Reset value	0			0		0	0	0	0	0	0	0	0	0	0	0	0				0	0	1		0	0	0	0	0	0	0	0
0x024	ICACHE_CRR1	HBURST	Res.	Res.	MSTSEL	Res.	REMAPADDR[31:21]											REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	BASEADDR[28:21]							
	Reset value	0			0		0	0	0	0	0	0	0	0	0	0	0	0				0	0	1		0	0	0	0	0	0	0	0
0x028	ICACHE_CRR2	HBURST	Res.	Res.	MSTSEL	Res.	REMAPADDR[31:21]											REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	BASEADDR[28:21]							
	Reset value	0			0		0	0	0	0	0	0	0	0	0	0	0	0				0	0	1		0	0	0	0	0	0	0	0
0x02C	ICACHE_CRR3	HBURST	Res.	Res.	MSTSEL	Res.	REMAPADDR[31:21]											REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	BASEADDR[28:21]							
	Reset value	0			0		0	0	0	0	0	0	0	0	0	0	0	0				0	0	1		0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

9 Radio system

9.1 Introduction

The 2.4 GHz RADIO is ultralow power, operating in the 2.4 GHz ISM band. It provides Bluetooth® Low Energy 1 Mbps coded, 1 Mbps, and 2 Mbps noncoded GFSK, and IEEE802.15.4 chip rate 2 Mchip/s, spreading mode DSSS, data rate 125 and 250 kbps O-QPSK-C modulation.

The 2.4 GHz RADIO is compliant with the Bluetooth and Zigbee® specifications, and radio regulations including ETSI EN 300 328, EN 300 440, EN 301 489-17, ARIB STD-T66, FCC CFR47 part 15 section 15.205, 15.209, 15.247, and 15.249, IC RSS-139 and RSS-210.

9.2 Main features

- Radio protocol:
  - Bluetooth® Low Energy
  - IEEE802.15.4
- Bluetooth® Low Energy features:
  - Device privacy and network privacy modes
  - Anonymous device address types
  - Advertising extension PDUs
  - Advertising channel index
  - Periodic advertising synchronous transfer
  - High duty cycle, nonconnectable advertising
  - Channel selection algorithm #2
  - Angle of arrival (AoA), angle of departure (AoD)
  - Channel sounding
  - Up to eight connections in any role in addition to advertiser and scanner roles
  - Audio connected isochronous streams
  - Audio broadcast isochronous streams
- IEEE 802.15.4 features:
  - Beacon management
  - 16-bit short and 64-bit IEEE addressing modes
  - PAN formation along with association and disassociation
  - Full handshake protocol for transfer reliability, frame validation, and acknowledgment frame delivery
  - IEEE802.15.4 2020 MAC for nonbeaconed PANs
- External PA support
- Packet traffic arbitration

9.3 2.4 GHz RADIO implementation

Table 81. Radio features

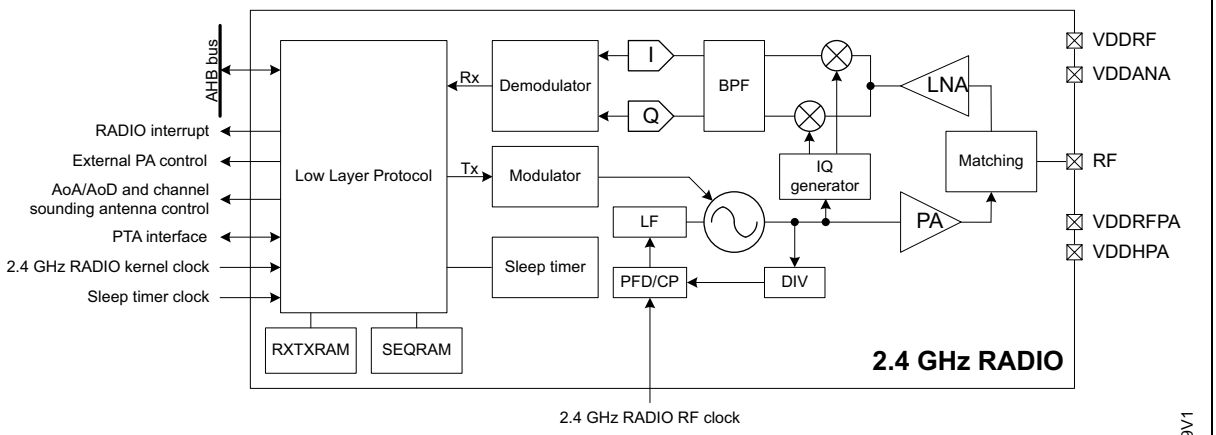
Feature[1]	STM32WBA 25xx	STM32WBA 23xx
Bluetooth (1 Mbps and codec PHY)	X	X
Bluetooth (2 Mbps)	X	X
Bluetooth AoA/AoD	X	X
IEEE802.15.4	X	X
Transmit maximum output power	+10 dBm	
External PA	X	X
Packet traffic arbitration	X	X

1. X = supported.

9.4 Functional description

9.4.1 Block diagram

Figure 23. Radio system block diagram



VDDANA is available only on packages with SMPS. On LDO packages VDDANA is double bonded with VDDRF.

9.4.2 Pins and internal signals

Table 82. Input/output pins

Pin name	Signal type	Description
RF	RF	2.4 GHz RF input output
RADIO interrupt	Output	2.4 GHz RADIO interrupt to the CPU
External PA control	Output	External PA control

Table 82. Input/output pins (continued)

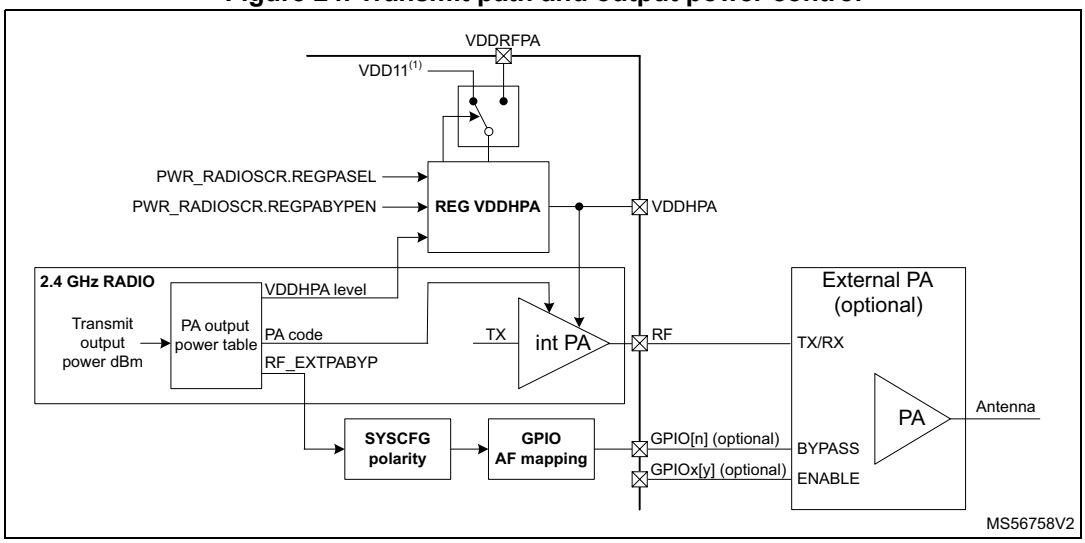
Pin name	Signal type	Description
AoA/AoD antenna control	Output	Angle of arrival/departure antenna area control
PTA interface	Input/Output	Packet traffic arbitration control

9.4.3 Transmit output power

The 2.4 GHz RADIO includes an internal PA. The transmit output power can be increased using an additional external PA. External PA control signal BYPASS can be selected on GPIOs (see the datasheet), and the signal ENABLE can be mapped on any free GPIO by the application software. The polarity of the BYPASS signal is selected in SYSCFG.

Figure 24 shows the transmit path and output power control.

Figure 24. Transmit path and output power control



1. VDD11 must be selected only when supplied from device SMPS.

The transmit output power (in dBm) is dependent upon the V_{DDHPA} voltage level and the PA code, and, when the application supports an external PA, an external PA bypass indication. The values for these parameters are defined in a PA output power table available to the Link Layer software.

Table 83. PA output power table format

V_{DDHPA} level	PA code	BYPASS
0x0 = 0.90 V 0x1 = 0.95 V 0x2 = 1.0 V 0x3 = 1.1 V ... 0xE = 2.2 V 0xF = 2.3 V	0 to 25 dec.	0 or 1

The internal PA maximum transmit output power is depended upon the V_{DDHPA} voltage level and the PA code. The maximum V_{DDHPA} voltage level is dependent upon the application (VDDRFPA pin) supply scheme, see *Table 84* for details. Note that the 2.4 GHz RADIO internal V_{CCRF} regulated supply level also depends upon the VDDRFPA pin supply scheme:

- If VDDRFPA < 1.71 V, the V_{CCRF} regulated supply level must be 1.0 V.
- If VDDRFPA ≥ 1.71 V, the V_{CCRF} regulated supply level must be 1.2 V.

The REG VDDHPA regulator input voltage can be controlled through REGPASEL and REGPABYPEN.

Table 84. 2.4 GHz RADIO supply configuration

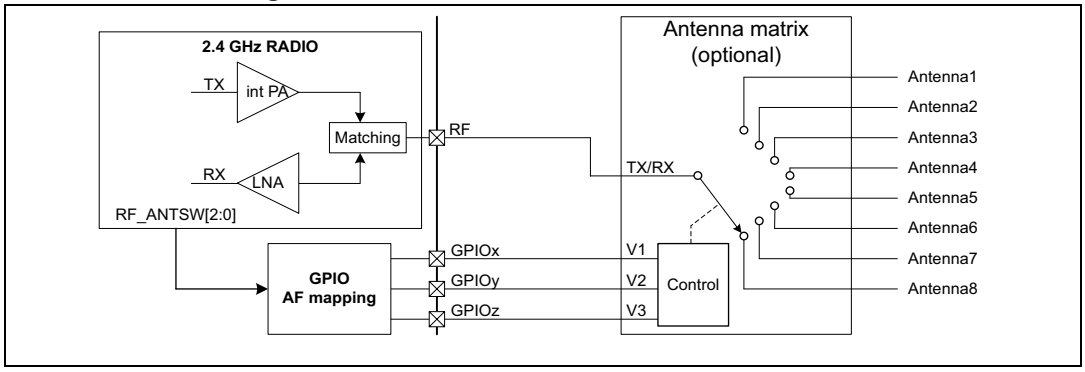
V_{DDHPA} level	Minimum V_{DDRFPA}	Internal PA max transmit output power
1.0 V	1.2 V(1)	+ 3 dBm
1.2 V	1.71 V	+ 6 dBm
1.5 V	1.8 V	+ 8 dBm
2.3 V	2.6 V	+ 10 dBm

1. Only on STM32WB25xx devices, when SMPS is used.

9.4.4 Bluetooth AoA/AoD and channel sounding antenna selection

Up to eight antennas are supported by means of the coded RF_ANTSW[2:0] signal bus available through GPIO alternate function for Bluetooth AoA when receiving the constant tone extension of a packet, or for AoD while transmitting the constant tone extension of a packet. Up to four antennas are supported with channel sounding.

Figure 25. Bluetooth AoA/AoD antennas control



9.4.5 RXTX data SRAM access

For the host application to access the 2.4 GHz RADIO RXTXRAM, the Link Layer software must put the 2.4 GHz RADIO in Sleep mode. When access to the 2.4 GHz RADIO RXTXRAM is no longer needed by the host application, it must request the Link Layer software to put the 2.4 GHz RADIO in Deepsleep mode.

To access the RXTX data SRAM, both the 2.4 GHz RADIO bus clock and kernel clock must be active and ready.

The 2.4 GHz RADIO modes depend both on the host application request and the Link Layer scheduled radio activity.

9.5 Low-power modes

The 2.4 GHz RADIO supports autonomous operation down to Stop 0 mode voltage scaling range 1 and 1.5, and the 2.4 GHz RADIO configuration and sleep timer remains operational down to Standby retention mode.

The 2.4 GHz RADIO in autonomous mode (Active mode)is able to transmit and receive packets over the air for the radio connection.

The 2.4 GHz RADIO configuration SRAMs and sleep timer allows to maintain radio connections when the radio is in low-power mode (Sleep and Deepsleep mode), this i.e. in between the Bluetooth LE connection events.

The 2.4 GHz RADIO must be disabled before entering Standby mode.

Table 85. Effect of low-power modes on the 2.4 GHz RADIO

Mode	Description
Sleep	No effect. 2.4 GHz RADIO (only in voltage scaling range 1) and 2.4 GHz RADIO sleep timer interrupts cause the device to exit Sleep mode. Radio connection are maintained and may transmit and receive.
Stop 0 voltage scaling range 1 and 1.5	The 2.4 GHz RADIO will autonomously transmit and receive packets, 2.4 GHz RADIO and 2.4 GHz RADIO configuration and sleep timer interrupts cause the device to exit Stop 0 mode. Radio connection are maintained and may transmit and receive.
Stop 0 voltage range 2 and Stop 1, and Stop 2	The 2.4 GHz RADIO configuration is retained but is no longer capable to transmit and receive packets, The 2.4 GHz RADIO sleep timer interrupt cause the device to exit Stop mode. Radio connection are maintained during i.e. the Bluetooth LE connection interval.
Stop 3	The 2.4 GHz RADIO and the 2.4 GHz RADIO sleep timer are no longer operational. Radio connection are lost.
Standby retention	The 2.4 GHz RADIO configuration is retained but is no longer capable to transmit and receive packets, The 2.4 GHz RADIO sleep timer interrupt cause the device to exit Standby retention mode. Radio connection are maintained during i.e. the Bluetooth LE connection interval.
Standby	The 2.4 GHz RADIO and 2.4 GHz RADIO sleep timer are powered down. Radio connection are lost.

10 PTA converter (PTACONV)

10.1 PTACONV introduction

The PTA converter is used to convert the PTA information coming from the 2.4 GHz RADIO into the PTA protocol used on the PTA controller GPIOs.

10.2 PTACONV main features

- Based on the IEEE802.15.2 standard
- Supports both grant and deny signaling
- Supports from 1- to 4-wire protocols
- Programmable transmit receive PTA_STATUS polarity
- Programmable priority polarity
- Programmable grant polarity
- Programmable active polarity
- Programmable PTA_ACTIVE timing
- Programmable PTA_STATUS time-multiplexed priority timing
- Programmable transmit packet abort

10.3 PTACONV functional description

The PTA converter adapts the PTA protocol to the various external PTA controllers. It supports a grant or deny signaling on the PTA_GRANT signal, and the following protocols:

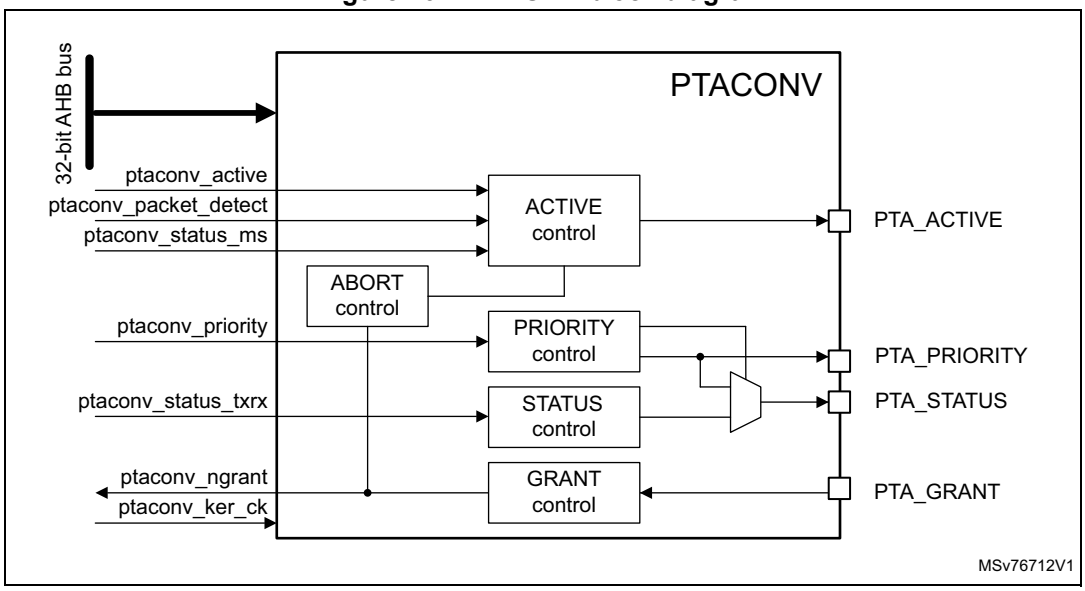
- 1-wire PTA_ACTIVE
- 2-wire PT_ACTIVE and PTA_GRANT
- 3-wire PTA_ACTIVE, PTA_GRANT, and time-multiplexed PTA_STATUS
- 4-wire PTA_ACTIVE, PTA_GRANT, PTA_PRIORITY and PTA_STATUS

When the request to cease an ongoing transmission occurs near the end of the 2.4 GHz RADIO packet, PTA_ACTIVE is deactivated at the end of the packet, before the expiration of T4.

10.3.1 PTACONV block diagram

Figure 26 shows the PTA converter block diagram and the interface to the 2.4 GHz RADIO.

Figure 26. PTACONV block diagram



The PTACONV is clocked by the 2.4 GHz RADIO kernel clock, independently from the hclk for the AHB interface used to access the PTACONV configuration registers.

10.3.2 PTACONV pins and internal signals

Table 86. 2.4 PTACONV input/output pins

Pin name	Signal type	Description
PTA_ACTIVE	Output	PTA 2.4 GHz RADIO packet activation request
PTA_PRIORITY	Output	PTA 2.4 GHz RADIO packet priority
PTA_STATUS	Output	PTA 2.4 GHz RADIO packet type
PTA_GRANT	Input	PTA grant medium to 2.4 GHz RADIO

Table 87. PTACONV internal input/output signals

Pin name	Signal type	Description
ptaconv_active	Input	2.4 GHz RADIO packet activation request
ptaconv_priority	Input	2.4 GHz RADIO packet priority
ptaconv_status_txrx	Input	2.4 GHz RADIO packet type Tx or Rx
ptaconv_status_ms	Input	2.4 GHz RADIO packet type maser or slave
ptaconv_packet_detect	Input	2.4 GHz RADIO packet detect
ptaconv_ngrant	Output	PTA active low medium granted

Table 87. PTACONV internal input/output signals (continued)

Pin name	Signal type	Description
ptaconv_ker_ck	Input	Kernel clock
AHB	Input/output	AHB bus interface

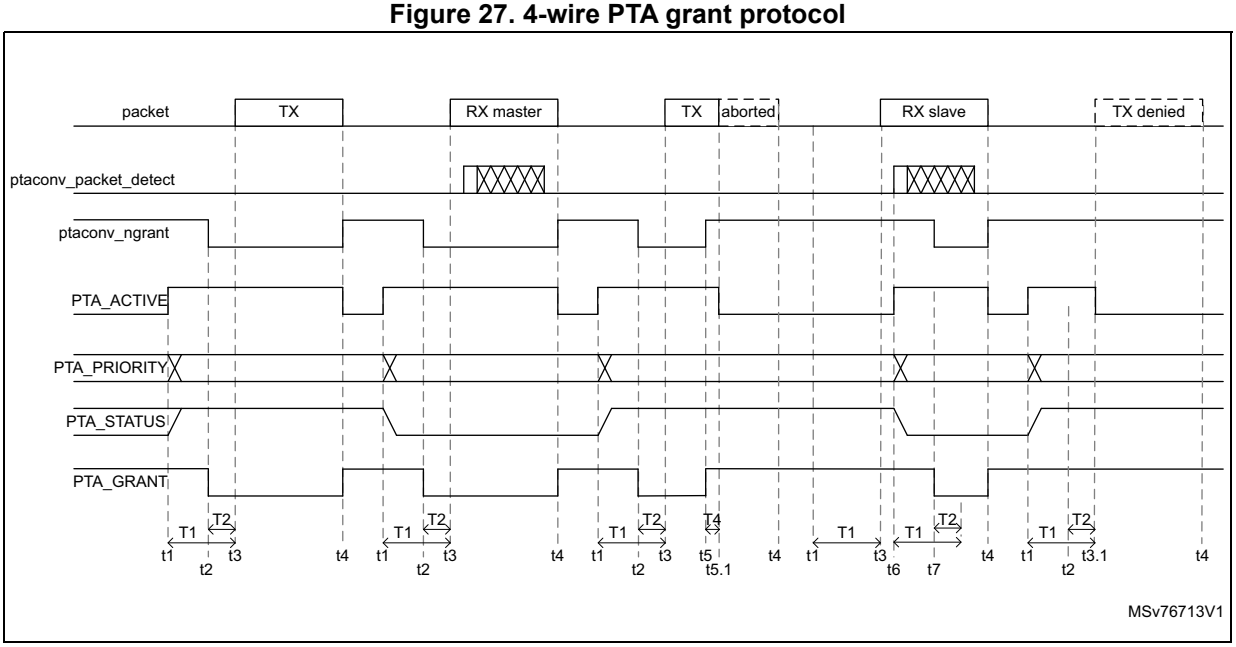
10.3.3 PTACONV protocols

The PTA grant protocol, the PTA deny protocol, and the 3-wire PTA_STATUS time-multiplexed priority and status transmit receive information are detailed in this section.

PTA grant protocol

The 4-wire PTA grant protocol uses PTA_ACTIVE, PTA_STATUS, PTA_PRIORITY, and PTA_GRANT signals, as shown in *Figure 27*.

Figure 27. 4-wire PTA grant protocol



Timing parameters:

- T1 is the time for the PTA_ACTIVE signal to request the medium before the start of the 2.4 GHz RADIO packet transfer.
- T2 is the time before the start of the 2.4 GHz RADIO packet transfer, during which the PTA_GRANT is stable.
- T4 is the delay time to cease an ongoing 2.4 GHz RADIO packet transmission and to deactivate PTA_ACTIVE.

Time instances:

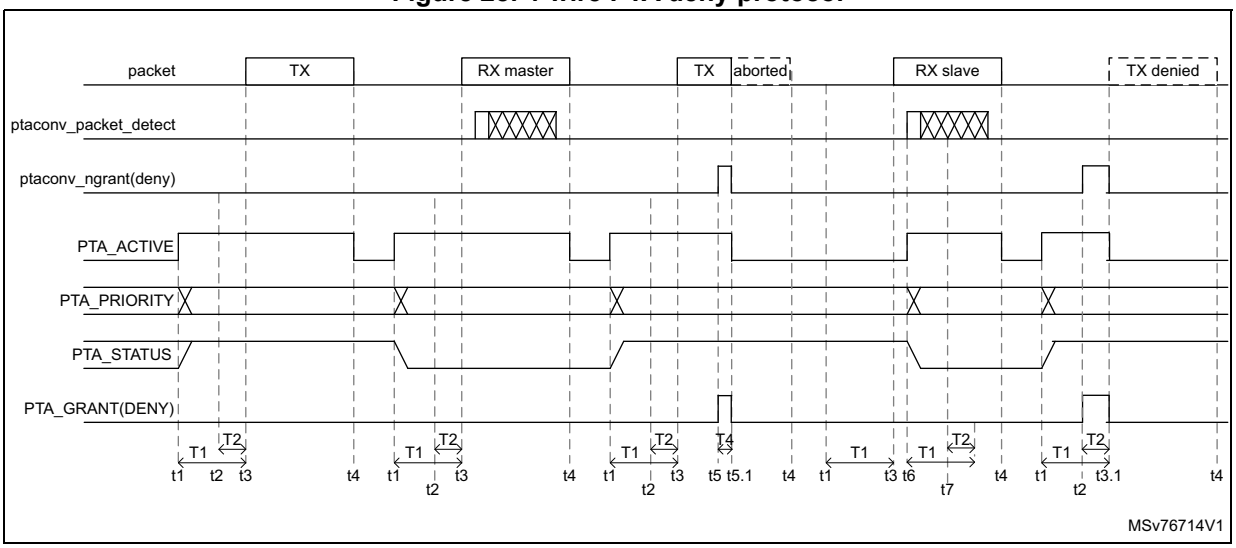
- t1: the request signal is activated before the 2.4 GHz RADIO packet transfer. For transmit and receive master packets, t1 is also the time PTA_ACTIVE is activated.
- t2: PTA_GRANT signal is stable.
- t3 start of the 2.4 GHz RADIO packet transfer.
- t3.1: the 2.4 GHz RADIO packet transfer is denied, and PTA_ACTIVE is deactivated. The external PTA controller can remove the PTA_GRANT as well.
- t4: indicates the end of the 2.4 GHz RADIO packet and the deactivation of the PTA_ACTIVE signal when active. The external PTA controller can remove the PTA_GRANT as well.
- t5: PTA_GRANT goes to deny and requests to cease the ongoing packet transmission.
- t5.1: indicates that the ongoing packet is aborted and the PTA_ACTIVE signal is deactivated. Abortion of the transmit packet can be disabled with register bit. ABORTDIS, in this case, the ngrant signal stays low until the end of the transmit packet.
- t6: the receive packet has been detected and used for slave receive packets to activate the PTA_ACTIVE signal.
- t7: PTA_GRANT is stable.

Receive packets are always granted and reception always proceeds. Receive packets use the PTA protocol to request the reservation of the medium for high priority receive packets.

PTA deny protocol

The PTA deny protocol uses PTA_ACTIVE, PTA_STATUS, PTA_PRIORITY and PTA_DENY signals, as shown in *Figure 28*.

Figure 28. 4-wire PTA deny protocol



Timing parameters:

- T1 is the time for the PTA_ACTIVE signal to request the medium before the start of the 2.4 GHz RADIO packet transfer.
- T2 is the time before the start of the 2.4 GHz RADIO packet transfer, during which the PTA_GRANT is stable.

Time instances:

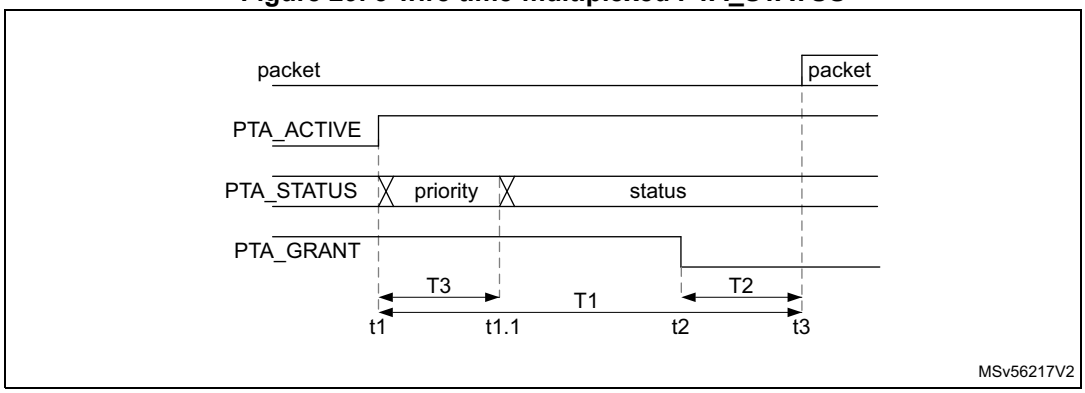
- t1: the request signal is activated before the 2.4 GHz RADIO packet transfer. For transmit and receive master packets, t1 is also the time PTA_ACTIVE is activated.
- t2: PTA_GRANT signal is stable.
- t3: start of the 2.4 GHz RADIO packet transfer.
- t3.1: a 2.4 GHz RADIO packet transfer is denied and the PTA_ACTIVE is deactivated. The external PTA controller can remove the PTA_GRANT as well.
- t4: indicates the end of the 2.4 GHz RADIO packet and the deactivation of the PTA_ACTIVE signal when active. The external PTA controller too can remove the PTA_GRANT.
- t5: PTA_GRANT goes to deny and requests to cease the ongoing packet transmission.
- t5.1: indicates that the ongoing packet is aborted and the PTA_ACTIVE signal is deactivated. Abortion of the transmit packet can be disabled with register bit ABORTDIS, in this case, the deny signal stays high until the end of the transmit packet.
- t6: the receive packet has been detected and used for slave receive packets to activate the PTA_ACTIVE signal.
- t7: PTA_GRANT is stable.

Receive packets are always granted and reception always proceeds. Receive packets use the PTA protocol to request the reservation of the medium for high priority receive packets.

3-wire time shared PTA_STATUS

The 3-wire PTA protocol does not use the PTA_PRIORITY signal. The priority and transmit receive packet status information is time-multiplexed on the PTA_STATUS signal, as shown in *Figure 29*.

Figure 29. 3-wire time-multiplexed PTA_STATUS



Timing parameters:

- T1 is the time for the PTA_ACTIVE to request the medium before the start of the 2.4 GHz RADIO packet transfer.
- T2 is the time before the start of the 2.4 GHz RADIO packet transfer during which the PTA_GRANT is stable.
- T3 is the time that the priority is time-multiplexed on the PTA_STATUS signal.

Time instances:

- t1: the request signal is activated before the 2.4 GHz RADIO packet transfer. For transmit and receive master packets, it is also the time PTA_ACTIVE is activated. Priority information is time-multiplexed on the PTA_STATUS signal.
- t1.1: the transmit or receive status is time-multiplexed on the PTA_STATUS signal.
- t2: PTA_GRANT signal is stable.
- t3: start of the 2.4 GHz RADIO packet transfer.

PTA timing parameters

Table 88. 2.4 PTACONV timing parameters

Symbol	Parameter	Min	Typ	Max	Unit
T1, T_{1_PTA}	PTA_ACTIVE setup time	20	-	150	µs
$T_{1_PTA_jitter}$	PTA_ACTIVE setup time jitter	-2	-	2	
T2, T_{2_PTA}	PTA_GRANT setup time	5	-	-	
T3, T_{3_PTA}	PTA_STATUS priority valid time	8	-	20	
T4, T_{4_PTA}	Transmit packet abort delay	5	-	10	

10.3.4 PTACONV interface with the 2.4 GHz RADIO

The 2.4 GHz RADIO provides the following signals:

- ptaconv_active:
  - activated with a T1 setup time before the start of the packet. This is before:
    The PA ramping for transmit packet
    The start of the receive window for receive packets
  - deactivated at the end of the scheduled packet. This is after:
    The PA has ramped down for transmit packets.
    The last receive packet bit has been received, or at the end of the review window when no packet is detected.
- ptaconv_priority: signaling 0 for low-priority packets, 1 for high priority packets
- ptaconv_status_txrx: signaling 0 for receive packets, 1 for transmit packets
- ptaconv_status_ms: signaling 0 for master packets, 1 for slave packets
- ptaconv_packet_detect: receive packet (access code) detect indication
- ptaconv_ngrant: active low grant signal
  - to be sampled during the valid window T2 to determine if a packet transmission is granted or denied
  - monitored during the transmit packet to cease transmission upon the reception of a deny

10.4 PTACONV low-power modes

The PTACONV supports operation in low power modes as shown in the table below:

Table 89. Effect of low-power modes on the PTACONV

Mode	Description
Sleep	No effect. PTACONV remains operational.
Stop 0 voltage scaling range 1 and 1.5	No effect. PTACONV remains operational.
Stop 0 voltage range 2 and Stop 1	The PTACONV register context is kept, PTA signals keep their values.
Stop 2, Stop 3	The PTACONV is powered down, PTA signals keep their values when enabled in register PWR_S2RETR.
Standby retention, Standby	The PTACONV is powered down, PTA signals keep their values when enabled in register PWR_IORETENRx.

10.5 PTACONV registers

10.5.1 PTACONV active control register (PTACONV_ACTCR)

Address offset: 0x000

Reset value: 0x0005 0014

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ABORT DIS	TABORT[3:0]			
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACTPOL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TACTIVE[7:0]							
rw								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **ABORTDIS**: Disable PTA_ACTIVE deny to abort an ongoing transmission
Set and cleared by software to define if PTA_ACTIVE aborts an ongoing transmission.
0: PTA_ACTIVE deny aborts an ongoing transmission
1: PTA_ACTIVE deny does not abort an ongoing transmission

Bits 19:16 **TABORT[3:0]**: PTA_ACTIVE delay to cease an ongoing transmission in µs
Set and cleared by software to define transmit packet abort delay and signaling on PTA_ACTIVE.
0x5 to 0xA: transmit packet PTA_ACTIVE abort delay time: T_{4_PTA} = TABORT × 1 µs
Others: Reserved, must no be used.

Bit 15 **ACTPOL**: PTA_ACTIVE polarity
Set and cleared by software to define PTA_ACTIVE signal polarity.
0: PTA_ACTIVE active high
1: PTA_ACTIVE active low

Bits 14:8 Reserved, must be kept at reset value.

Bits 7:0 **TACTIVE[7:0]**: PTA_ACTIVE setup time in µs
Set and cleared by software to define PTA_ACTIVE setup time.
0x14 to 0x96: PTA_ACTIVE setup time: T_{1_PTA} = TACTIVE × 1 µs
Others: Reserved, must no be used.

10.5.2 PTACONV priority control register (PTACONV_PRICR)

Address offset: 0x004

Reset value: 0x0000 000A

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIPOL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TPRIORITY[4:0]				
rw											rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PRIPOL**: Priority polarity

Set and cleared by software to define PTA_PRIORITY and time-multiplexed priority on PTA_STATUS signal polarity.
0: priority on PTA_PRIORITY or PTA_STATUS not inverted
1: inverted priority on PTA_PRIORITY or PTA_STATUS

Bits 14:5 Reserved, must be kept at reset value.

Bits 4:0 **TPRIORITY[4:0]**: Priority valid time in µs

Set and cleared by software to define PTA_STATUS signal priority valid time.
0x00: no time-multiplexed priority information on PTA_STATUS
0x08 to 0x14: priority information multiplexed on PTA_STATUS with valid time:
T_{3_PTA} = TPRIORITY × 1 µs
Others: Reserved, must no be used.

10.5.3 PTACONV control register (PTACONV_CR)

Address offset: 0x008

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRANT POL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXRX POL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

Bit 31 **GRANTPOL**: PTA_GRANT polarity

Set and cleared by software to define PTA_GRANT signal polarity.
0: PTA_GRANT active low
1: PTA_GRANT active high

Bits 30:16 Reserved, must be kept at reset value.

Bit 15 **TXRXPOL**: PTA_STATUS transmit and receive polarity

Set and cleared by software to define PTA_STATUS signal polarity for transmit and receive information.

0: PTA_STATUS receive = 0, transmit = 1

1: PTA_STATUS receive = 1, transmit = 0

Bits 14:0 Reserved, must be kept at reset value.

10.5.4 PTACONV register map

Table 90. PTACONV register map and reset values

Offset	Register	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
0x000	PTACONV_ ACTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ABORTDIS	TABORT[3:0]				ACTPOL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TACTIVE[7:0]							
	Reset value												0	0	1	0	1	0								0	0	0	1	0	1	0	0
0x004	PTACONV_ PRICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIPOL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TPRIORITY[4:0]				
	Reset value																	0											0	1	0	1	0
0x008	PTACONV_ CR	GRANTPOL	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TXRXPOL	Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
	Reset value	0																0															
0x00C to 0x3FC	Reserved	Reserved																															

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

11 Power control (PWR)

11.1 Introduction

The power controllers manages all device power supplies and power modes transitions.

11.2 PWR main features

The power controller (PWR) main features are:

- Power supplies and supply domains
  - Core domain (V_{CORE})
  - V_{DD} domain
  - Analog domain (V_{DDA})
  - Supply for the SMPS power stage
  - Supply for the 2.4 GHz RADIO
  - V_{DDUSB} domain for USB (available only on STM32WBA25xx)
- System supply voltage regulation
  - SMPS step-down converter
  - Linear voltage regulator (LDO)
- Power supply supervision
  - BOR monitor (including power-on reset)
  - PVD monitor
- Power management
  - Operating modes
  - Voltage scaling control
  - Low-power modes
- GPIO retention in Standby
- TrustZone® security and privileged protection

11.3 PWR pins and internal signals

Table 91. PWR input/output pins

Pin name	Signal type	Description
VDD	Supply	Main and backup domain supply
VDDUSB(1)	Supply	Independent USB supply
VDDA	Supply	Analog peripherals supply
VDDRF	Supply	2.4 GHz RADIO RF supply
VDDRFPA	Supply	2.4 GHz RADIO PA regulator supply
VDDANA(2)	Supply	2.4 GHz RADIO analog supply
VDDHPA	Output	2.4 GHz RADIO PA regulator output
VSS	Supply	Ground
VSSRF	Supply	2.4 GHz RADIO ground
VDD11	Input/Output	Logic supply (V_{CORE})
VCAP(2)	Output	Logic supply (V_{CORE})
VDDSMPS	Supply	SMPS supply
VSSSMPS	Supply	SMPS ground
VLXSMPS	Supply	SMPS output

1. Available only on STM32WBA25xx devices.
2. Available only on STM32WBA23xx devices.

Table 92. PWR internal input/output signals

Internal signal name	Signal type	Description
WKUPx_y (x = 1 to 8, y = 1 to 4)	Input	Wake-up event source selection
WKUP interrupt	Output	Global WKUP pin interrupt
SLEEP	Output	CPU in Sleep mode, device Sleep mode
STOP	Output	CPU in Deepsleep mode, device Stop mode

Each of the wake-up event WKUPx can be generated from device pins or internal events, selected by WUSELx[1:0] in the PWR_WUCR3 register (x = 1 to 8). A WKUP interrupt is generated only when WKUPx is generated from a device pin. WKUPx generated from internal events are associated with an internal event interrupt.

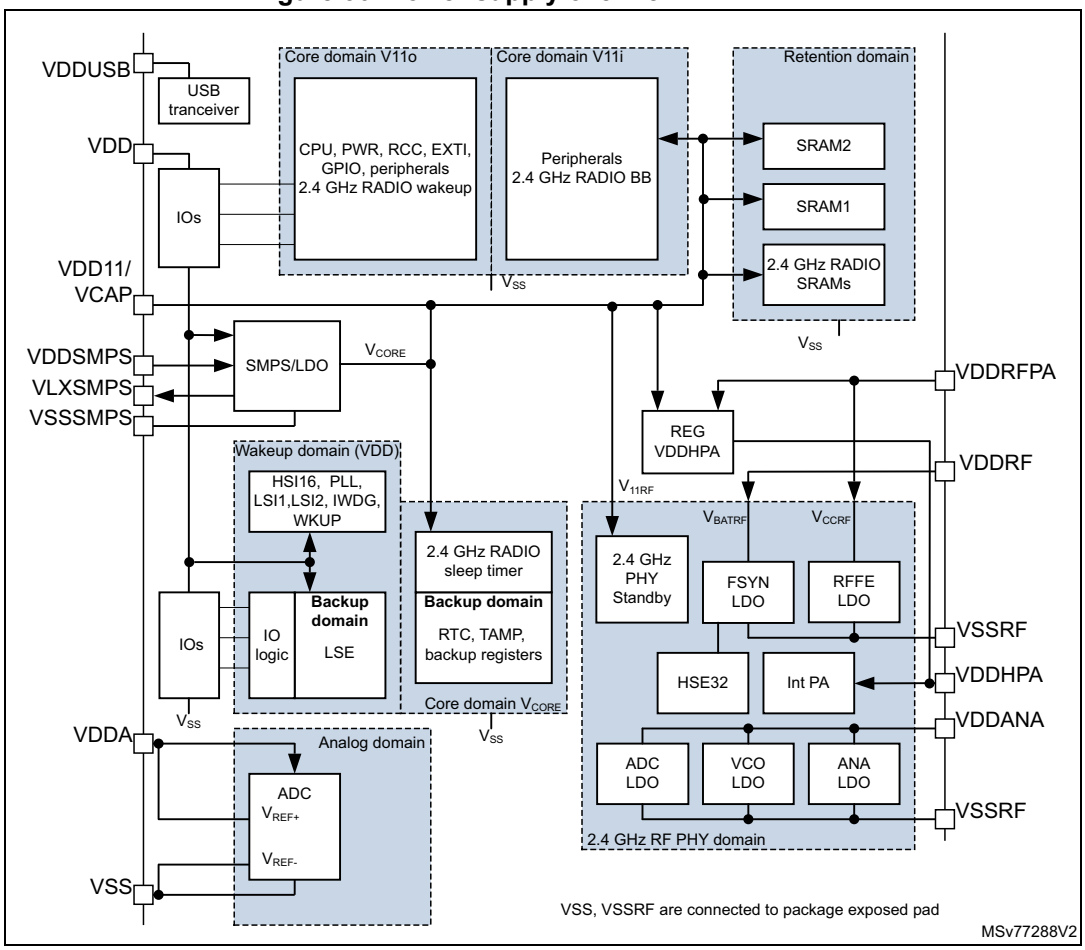
Table 93. PWR wake-up source selection

Wake-up event	Internal signal source (x = 1 to 8)			
	WKUPx_0 (WUSELx = 00)	WKUPx_1 (WUSELx = 01)	WKUPx_2 (WUSELx = 10)	WKUPx_3 (WUSELx = 11)
WKUP1	PA0(1)	PB2	Reserved	Reserved
WKUP2	Reserved	PC13	Reserved	Reserved
WKUP3	Reserved	PA1	PB6	Reserved
WKUP4	PA2	Reserved	Reserved	Reserved
WKUP5	Reserved	Reserved	Reserved	Reserved
WKUP6	PA12	PA5	Reserved	Secure rtc_alra_trg, rtc_alrb_trg, rtc_wut_trg, or RTC_TS
WKUP7	Reserved	PA6	Reserved	Nonsecure rtc_alra_trg, rtc_alrb_trg, rtc_wut_trg, or RTC_TS
WKUP8	Reserved	PA7	PB9	Any secure or nonsecure tamp_trg

1. Reserved on STM32WBA25xx devices.

11.4 PWR power supplies and supply domains

Figure 30. Power supply overview



Warning: When SMPS devices are used in an LDO application, without inductor between VLXSMPS and VDD11, the pins VDDSMPS and VLXSMPS must be connected to the ground.

11.4.1 External power supplies

The devices require a 1.71 V to 3.6 V V_{DD} operating voltage supply. Several independent supplies can be provided for specific peripherals. Those supplies must not be provided without a valid operating supply on the VDD pin:

- V_{DD} = 1.71 V to 3.6 V

  V_{DD} is the external power supply for the I/Os, the internal regulator, and the system analog such as reset, power management, and internal clocks. It is provided externally through the VDD pins.
- V_{DDUSB} = 3.0 V to 3.6 V (only available on STM32WAB25xx)

  V_{DDUSB} is the independent external power supply for the USB GPIO port PB[9:8] I/Os. When no independent supply is used, VDDUSB shall be connected to V_{DD}.
- V_{DDA} = 1.62 V to 3.6 V

  V_{DDA} is the external analog power supply for ADC. The V_{DDA} voltage level is independent from the V_{DD} voltage, and must preferably be connected to VDD when the peripherals are not used.
- V_{DDSMPS} = 1.71 V to 3.6 V

  V_{DDSMPS} is the external power supply for the SMPS step-down converter. It is provided externally through V_{DDSMPS} supply pin, and must be connected to the same supply as VDD pin.
- V_{LXSMPS} is the switched SMPS step-down converter output .

Note: The SMPS power supply pins are available only on specific package with SMPS step-down converter option.

- V_{DD11} / V_{CAP} = 0 V to 1.2 V
  V_{CORE} is the internal power supply for the digital logic.
- V_{DDRFPA} = 0 V to 3.6 V (need to be above 1.2 V for 2.4 GHz RADIO operation). This supply must be equal or lower than V_{DDRF}.

  V_{DDRFPA} is the external power supply for the 2.4 GHz RADIO front-end part and PA.
  - V_{DDRFPA} when connected to VDD11 supports a transmit maximum low output power.
  - V_{DDRFPA} when connected to VDD supports a transmit maximum high output power.
- V_{DDANA} = 0 V to 3.6 V (need to be above 1.2 V for 2.4 GHz RADIO operation). This supply must be equal or lower than V_{DDRF}.

  V_{DDANA} is the external power supply for the 2.4 GHz RADIO part, can be connected to VDD11 when REGPASEL select VDD11.
- V_{DDRF} = 1.71 V to 3.6 V

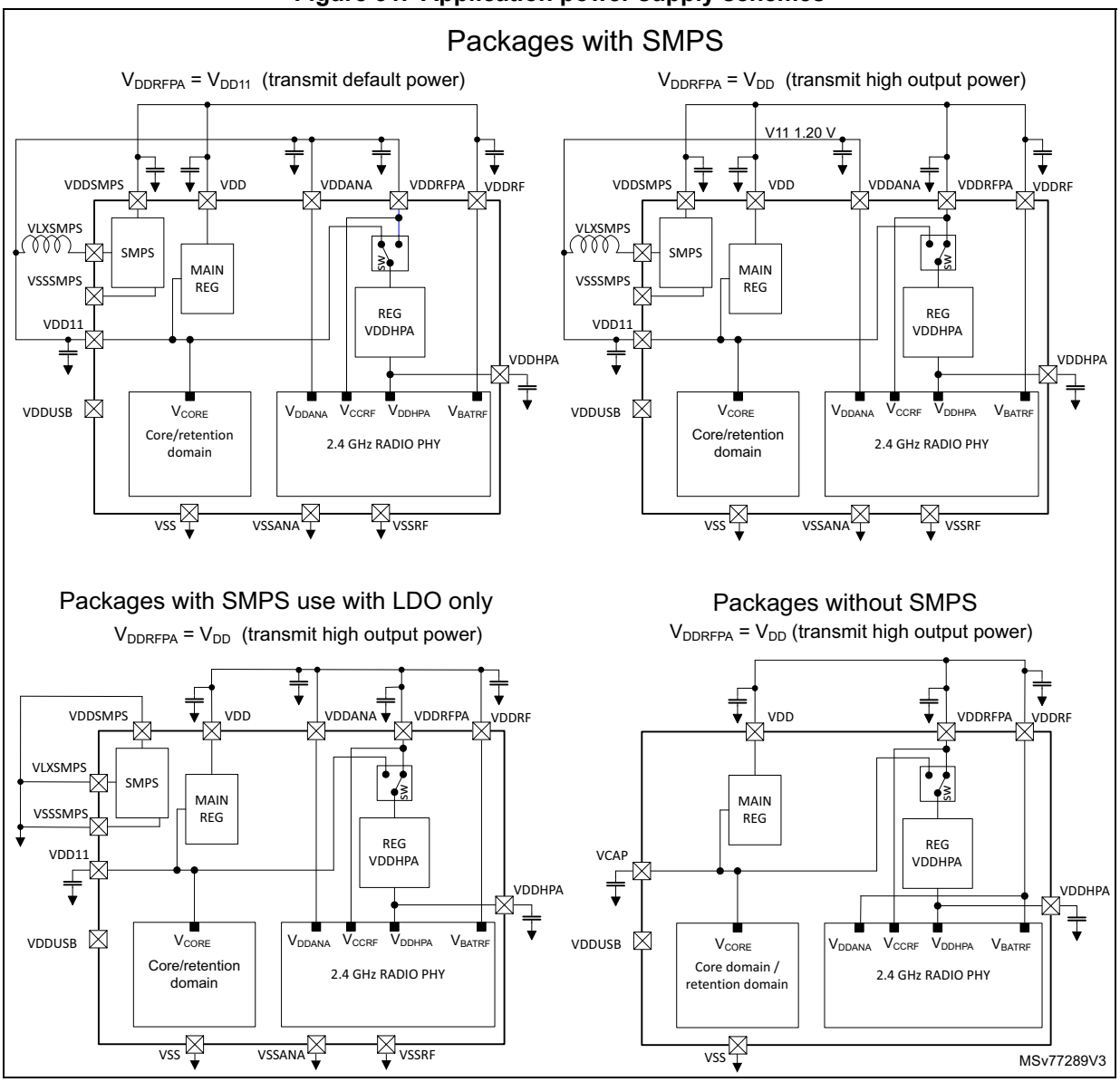
  V_{DDRF} is the external power supply for the 2.4 GHz RADIO. VDDRF must always be connected to the supply used for VDD.
- V_{DDHPA} = 0.9 V to 2.3 V
  V_{DDHPA} is the internal power supply for the 2.4 GHz RADIO power amplifier.

11.4.2 Application power supply schemes

The device supports different supply schemes, as shown in *Figure 31*. The maximum achievable transmit output power from the 2.4 GHz RADIO internal PA is linked to the supply scheme.

- In packages supporting an SMPS, a maximum transmit low output power can be reached with the lowest power consumption. In this power scheme V_{DDRFPA} is connected to V_{DD11}, and REGVDDHPA is connected to VDD11 selected internally by the REGPASEL register bit.
- In all other power schemes V_{DDRFPA} is connected to V_{DD}, providing maximum output power on transmit.

Figure 31. Application power supply schemes



11.4.3 Power-up and power-down power sequences

During power-up and power-down phases, the following power sequence requirements must be respected:

- When V_{DD} is below 1 V, other power supplies (V_{DDA}, V_{DDUSB}) must remain below V_{DD} + 300 mV.
- When V_{DD} is above 1 V, all power supplies are independent.

During the power-down phase, V_{DD} can temporarily become lower than other supplies only if the energy provided to the MCU remains below 1 mJ. This allows external decoupling capacitors with different time constants to be discharged during the power-down transient phase.

11.4.4 Independent analog peripherals supply

To improve A/D conversion accuracy and 2.4 GHz RADIO performance and to extend the supply flexibility, the analog and 2.4 GHz RADIO peripherals have independent power supplies that can be separately filtered and shielded from noise on the PCB:

- The analog peripheral voltage supply input is available on a separate VDDA pin
- The 2.4 GHz RADIO peripheral voltage supply input is available on a separate VDDRF, VDDANA, and VDDRFPA pin.

The V_{DDA} supply voltage can be different from V_{DD}. The V_{DDA} must be present before enabling the analog peripheral supplied by V_{DDA} (ADC4).

When a single supply is used, VDDA can be externally connected to VDD through the external filtering circuit in order to ensure a low-noise V_{DDA} voltage.

11.4.5 Independent USB transceiver supply

The USB transceiver is supplied from a separate VDDUSB supply pin. For USB operation the V_{DDUSB} range is from 3.0 V to 3.6 V independent from V_{DD} and other supplies.

After reset, the USB transceiver supplied by V_{DDUSB} is logically and electrically isolated and therefore is not available. The isolation must be removed before using the USB peripheral by setting the USV bit in the PWR_SVMCR register, once the V_{DDUSB} supply is present.

There is no dedicated V_{DDUSB} supply monitor. The application software must ensure that V_{DDUSB} is supplied at the right level before removing the logical and electrical isolation. Ensuring that V_{DDUSB} is supplied can be done by a software time delay, or measuring V_{DDUSB} using an ADC channel.

11.4.6 Independent RADIO peripheral supply

The 2.4 GHz RADIO V_{DDRF} and V_{DDRFPA} supply voltages can be different from V_{DD}. The 2.4 GHz RADIO supplies must be present before enabling the 2.4 GHz RADIO.

When a single supply is used, VDDRF and VDDRFPA supplies can be externally connected to VDD through external filtering circuitry to ensure noise-free supply voltages.

11.4.7 Backup domain

The backup domain contains the RTC, TAMP, backup registers supplied from V_{CORE}, and the LSE oscillator directly supplied from VDD pin.

Backup domain access

After a system reset, the backup domain (RCC backup domain control register RCC_BDCR1, RTC registers, TAMP registers, backup registers) is protected against possible unwanted write accesses. To enable access to the backup domain, proceed as follows:

1. Enable the power interface clock by setting the PWREN bits in the *RCC AHB4 peripheral clock enable register (RCC_AHB4ENR)*.
2. Set the DBP bit in the *PWR disable backup domain register (PWR_DBPR)* to enable access to the backup domain.

11.4.8 Internal regulators

The devices embed two regulators: one LDO and one SMPS in parallel to provide the V_{CORE} (for digital peripherals, SRAMs and embedded flash memory). Both regulators generate this voltage on VDD11 / VCAP pin. The SMPS is available only on some devices.

Both regulators can provide different voltage scaling ranges (voltage scaling) and can operate in Run and Stop modes.

It is possible to switch from SMPS to LDO and from LDO to SMPS and change voltage scaling range on the fly (when the 2.4 GHz RADIO is not active).

Other internal supplies for the 2.4 GHz RADIO are generated for the dedicated regulator.

11.5 PWR system supply voltage regulation

11.5.1 SMPS and LDO embedded regulators

All devices embed an internal linear voltage regulator (LDO). Some devices additionally embed an internal SMPS step-down converter, which can be selected when the application runs, depending on the application requirements.

The SMPS allows the power consumption to be reduced. Some peripherals can be affected by the noise generated by the SMPS, requiring the application to switch to LDO when running this peripheral, to reach the best performances.

The LDO or SMPS can be used in all voltage scaling ranges, and in all Stop modes and Standby with retention mode.

11.5.2 LDO and SMPS versus reset, voltage scaling, and low-power modes

After BOR0 power-on reset and system reset the LDO regulator is enabled in voltage scaling range 2. Switching to the SMPS regulator provides lower consumption in particular at high V_{DD} voltage level. It is possible to switch from LDO to SMPS, or from SMPS to LDO in any voltage scaling range, by configuring the REGSEL register bit.

When exiting from Stop or Standby retention modes, the regulator is the same than when entering these low-power modes. When exiting from Standby modes, the LDO regulator is always used to startup. When Standby has been entered from the SMPS regulator, after exiting Standby with the LDO, the regulator is then switched automatically to the SMPS regulator.

When exiting from Stop 0 modes the voltage scaling range is the same as on entering Stop 0 mode. When exiting from Stop 1, Stop 2, Stop 3, and Standby modes the voltage scaling range 2 is used.

When the 2.4 GHz RADIO is active, the regulator and voltage scaling range cannot be changed. Any requested regulator or voltage scaling range change while the 2.4 GHz RADIO is active, is suspended and takes effect only after the 2.4 GHz RADIO and PHY have entered Sleep or Deepsleep mode.

11.5.3 LDO and SMPS fast startup

After BOR0 power-on reset, the LDO regulator starts in high-power mode and in slow-startup mode. The slow-startup feature is selected to limit the inrush current after power-on reset. This increases the wake-up time also when exiting Standby modes.

It is possible to configure fast-startup on the fly and it is applied for the next startup either after a system reset or wake-up from standby modes. The fast-startup is selected by setting the FSTEN bit in the PWR_CR3 register. Fast-startup selection applies to both LDO and SMPS regulators.

11.5.4 Dynamic voltage scaling management

The dynamic voltage scaling is a power management technique that consists in increasing or decreasing the voltage used for the digital peripherals (V_{CORE}), according to the application performance and power consumption needs.

Dynamic voltage scaling to increase V_{CORE} is known as overvolting. It allows the device to improve its performance.

Dynamic voltage scaling to decrease V_{CORE} is known as undervolting. It is performed to save power, particularly in devices where the energy comes from a battery and is thus limited.

The regulator operates in the following voltage scaling ranges:

- Range 1 and range 1.5: high performance

  It allows a system clock frequency up to 64 MHz, and is required for any 2.4 GHz RADIO transmit and receive operation.

  When the 2.4 GHz RADIO is active the range cannot be changed. Any requested voltage scaling range change while the 2.4 GHz RADIO is active, is suspended and only takes effect after the 2.4 GHz RADIO has entered Sleep or Deepsleep mode.

  Range 1.5 is used to run the system up to 64 MHz at lower power consumption.

- Range 2: low-power

  The system clock frequency can be up to 16 MHz. The 2.4 GHz RADIO cannot transmit nor receive.

Voltage scaling is selected through the VOS bit in the PWR_VOSR register.

The sequence to switch the voltage scaling from range 2 to range 1 or range 1.5 is the following:

1. Program the VOS to range 1 or range 1.5 in the PWR_VOSR
2. Wait until the VOSRDY flag is set in the PWR_VOSR
3. Enable the HSE32 oscillator and/or PLL source
4. Adjust the number of flash wait states according to the new target SYSCLK frequency using flash LATENCY bitfield in FLASH_ACR.
5. Switch SYSCLK to HSE32 or pll1rclk.

The sequence to switch the voltage scaling from range 1 or range 1.5 to range 2 is the following:

1. Switch to the SYSCLK frequency ≤ 16 MHz, HSI16 or HSE32 divided by 2.
2. Wait for the frequency switch to be ready.
3. Adjust the number of wait states according to the new target SYSCLK frequency using flash LATENCY bitfield in FLASH_ACR, and SRAMs WSC bitfield in RAMCFG_MxCR.
4. Disable the PLL1
5. Program the VOS to range 2 in the PWR_VOSR
6. Optionally wait until the ACTVOS in the PWR_SVMSR = VOS in the PWR_VOSR and ACTVOSRDY flag is set in PWR_SVMSR

Note: When switching the voltage scaling, the sequence must be completed (VOS = ACTVOS and ACTVOSRDY = 1) before entering Stop, or Standby modes. When the 2.4 GHz RADIO is active (PWR_RADIOSCR.MODE = active) the system does not enter low-power mode and the CPU can enter Deepsleep independently from VOS, ACTVOS and ACTVOSRDY.

11.5.5 2.4 GHz RADIO PA regulator

The PA regulator REG VDDHPA is used to supply the 2.4 GHz RADIO internal PA with the correct voltage level V_{DDHPA} for a given transmit output power. This regulator output voltage V_{DDHPA} is controlled from the 2.4 GHz RADIO link layer software. The V_{DDHPA} maximum supply level and associated maximum transmit output power is depended on the application supply scheme. See *Section 11.4.2: Application power supply schemes*.

The application software can control the REG VDDHPA regulator input voltage.

- Force the input voltage selection of the REG VDDHPA regulator to come from VDDRFPA pin by the REGPASEL register bit.
  - When the REG VDDHPA regulator input voltage is forced to VDDRFPA, the power consumption at low transmit output power is higher.
- Allow the 1.2 V V_{DDHPA} voltage level to be generated directly from the VDD11 supply via the REGPABYPEN register bit. This can be selected only when VDD11 is supplied from the device SMPS. This is only available when the input voltage of the REG VDDHPA regulator is not forced to VDDRFPA.
  - When the SMPS is used, allowing the 1.2 V V_{DDHPA} voltage level to be generated directly from the VDD11 lowers the power consumption at the transmit output levels using this supply level.

The REG VDDHPA is forced off, discharging the external capacitor, in Stop 2, 3, and all Standby modes.

For more information on the 2.4 GHz RADIO transmit output power and REG VDDHPA regulator control, see *Section 9.4.3: Transmit output power*.

11.6 PWR power supply supervision

11.6.1 Brownout reset (BOR)

The device has an integrated BOR (brownout reset) circuitry. The BOR is active in all power modes, and cannot be disabled. The BOR reset also generates a system reset on pin NRST.

Five BOR thresholds can be selected through option bytes.

A power-on reset is always generated at the V_{BOR0} thresholds.

The backup domain supplied by V_{DD} is reset by the V_{BOR0} thresholds.

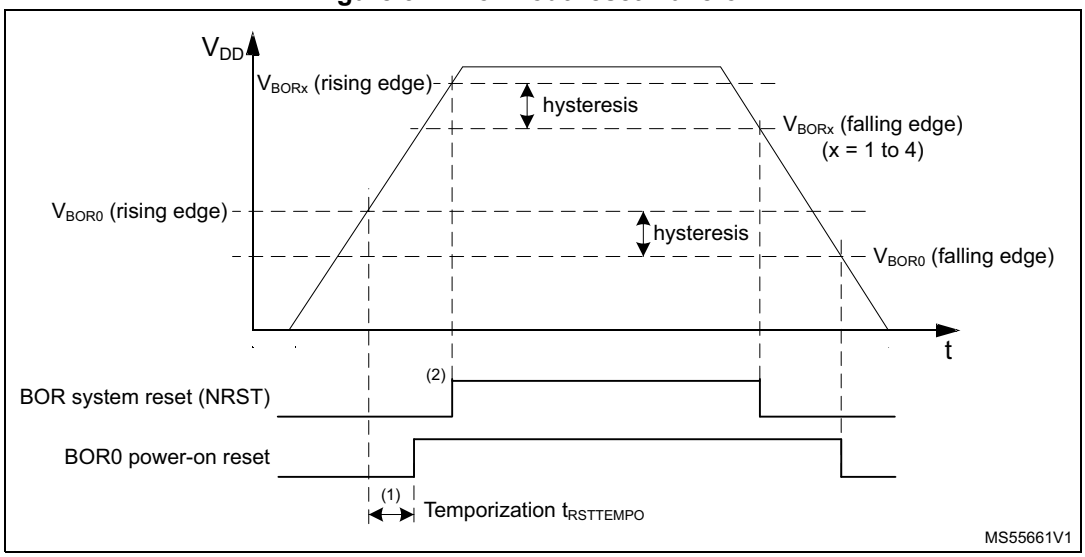
The other V_{BORx} (x = 1 to 4) thresholds keep most of the device under system reset until the supply voltage V_{DD} reaches the specified V_{BORx} threshold. When V_{DD} drops below the selected threshold, a system reset is generated on pin NRST. When V_{DD} is above the V_{BORx} upper limit, the system reset on pin NRST is released and the system can start.

For more details on the brownout reset thresholds, refer to the electrical characteristics section in the datasheet.

During Stop 1, 2, 3 and Standby modes, it is possible to set the BOR0 in ultra-low-power (discontinuous) mode to further reduce the current consumption by setting the ULPMEN bit in *PWR control register 1 (PWR_CR1)*.

Warning: In ultralow-power mode, the supply monitoring is discontinuous and can not detect fast drops in supply, hence it must not be used together with autonomous peripherals using HSI16 as kernel clock.

Figure 32. Brownout reset waveform



1. The reset temporization $t_{RSTTEMPO}$ is present only for the BOR0 lowest threshold (V_{BOR0}).
2. The rising edge of the system reset NRST can be driven from the V_{BOR0} power-on reset or V_{BORx} (x = 1 to 4) rising threshold depending on the V_{DD} rising slope.

11.6.2 Programmable voltage detector (PVD)

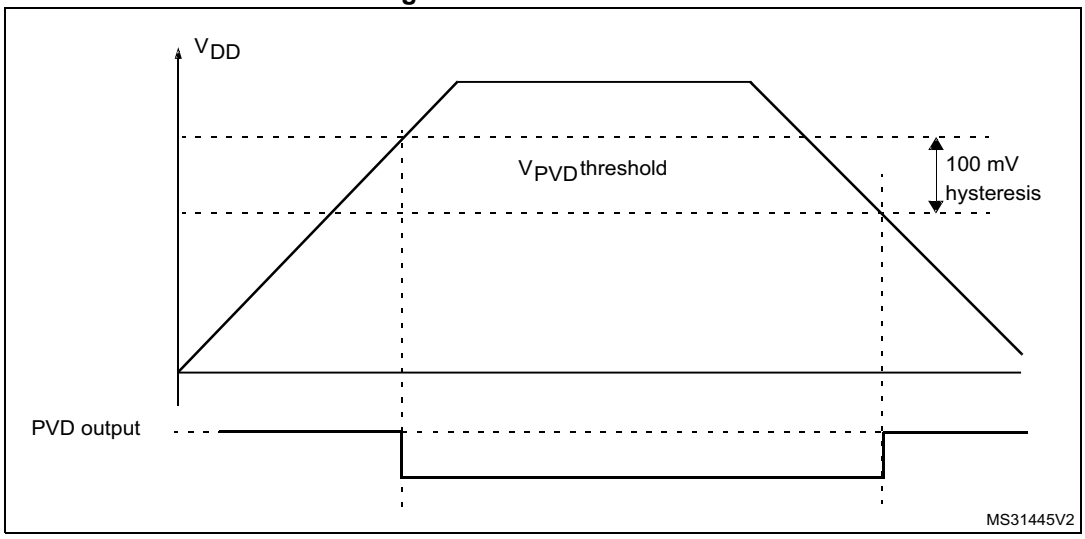
The PVD can be used to monitor the V_{DD} power supply by comparing it to a threshold selected by the PVDLS[2:0] bits in the *PWR supply voltage monitoring control register (PWR_SVMCR)*. The PVD is enabled by setting the PVDE bit.

A PVDO flag is available in the *PWR supply voltage monitoring status register (PWR_SVMSR)* to indicate if V_{DD} is higher or lower than the PVD threshold. This event is internally connected to the EXTI and can generate an interrupt, if enabled through the EXTI registers (refer to *Table 105: PWR interrupt requests*).

The rising/falling edge sensitivity of the EXTI line must be configured according to the PVD output behavior. For example, if the EXTI line is configured to rising edge sensitivity, the interrupt is generated when V_{DD} drops below the PVD threshold. As an example the service routine can perform emergency shutdown tasks.

The PVD can remain active in Stop modes, and the PVD interrupt can wake up the system from Stop mode. The PVD is not functional in Standby mode.

Figure 33. PVD thresholds



11.7 PWR power management

11.7.1 PWR power modes

By default, after a system or a power reset, the microcontroller is in Run mode voltage scaling range 2 and the 2.4 GHz RADIO is in Deepsleep. Several low-power modes are available to save power when the CPU or peripherals do not need to be kept running (for example when waiting for an external event). It is up to the user to select the mode that gives the best compromise between low-power consumption, short startup time, and available wake-up sources.

The device features these low-power modes:

- Run mode:
  The CPU is running.

  The power consumption in Run mode can be reduced by one of the following means:
  - Slowing down the system clocks and configuring voltage scaling to lower-power range.
  - Gating the clocks to the APB and AHB peripherals when they are unused.
- Sleep mode:

  CPU clock off, all peripherals including Cortex-M33 core, NVIC, and SysTick can run and wake up the CPU when an interrupt or an event occurs. Refer to *Section 11.7.5: PWR Sleep mode*.
- Stop 0, Stop 1 modes:

  The Stop modes achieve the lowest power consumption while retaining the content of registers. The SRAM content can be selected to be retained or not. All clocks in the core domain are stopped (except some autonomous peripheral bus and kernel clocks). The PLL1 and HSE32 crystal oscillators are disabled. The HSI16 RC oscillator is disabled when not activated as autonomous peripheral bus or kernel clock. The LSE or LSI can still run.

  Some peripherals are autonomous and can operate in Stop modes by requesting their kernel clock, and also in Stop 0 mode when requesting their bus clock as needed. When a peripheral request its (APB or AHB) bus clock, for example in order to transfer data with LPDMA1, the device transitions from Stop 1 to Stop 0 mode.

  The I2C, USART, LPUART, SPI, LPTIM, ADC, RTC, TAMP, IWDG can remain active in Stop modes when the kernel clock is LSE, LSI, or HSI16. The 2.4 GHz RADIO can remain active in Stop 0 mode on its kernel clock HSE32. In Stop 0 mode the autonomous peripheral bus clock can remain active using HSI16.

  The brownout reset (BOR) always remains active and the PVD can be kept active in Stop modes. In Stop 1 mode the BOR0 can be configured in ultralow-power mode to further reduce power consumption.

Warning: Ultralow-power mode must not be used together with autonomous peripherals using HSI16 as kernel clock.

  In Stop 0 mode, the regulator remains in main regulator mode, allowing a very fast wake-up time but with much higher consumption than Stop 1.

  When exiting from Stop mode, the system clock is HSI16 at 16 MHz.

  Refer to *Section 11.7.6: PWR Stop 0 mode*, and *Section 11.7.7: PWR Stop 1 mode*.
- Stop 2 mode:

  The Stop 2 mode achieves lower power consumption while retaining some peripherals and the system configuration registers content. The SRAM content can be selected to be retained or not. All clocks in the core domain are stopped (except some autonomous peripheral kernel clocks). The PLL and HSE32 crystal oscillators are disabled. The HSI16 RC oscillator is disabled when not activated as autonomous peripheral kernel clock. The LSE or LSI can still run.

Some peripherals are autonomous and can operate in Stop 2 mode by requesting their kernel clock as needed.

The I2C3, LPUART1, SPI3, LPTIM1, RTC, TAMP, IWDG, and 2.4 GHz RADIO sleep timer can remain active in Stop 2 mode when the kernel clock is LSE, LSI, or HSI16.

The brownout reset (BOR) always remains active and the PVD can be kept active in Stop 2 mode. The BOR0 can be configured in ultralow-power mode to further reduce power consumption.

Warning: Ultralow-power mode must not be used together with autonomous peripherals using HSI16 as kernel clock.

When exiting from Stop mode, the system clock is HSI16 at 16 MHz.

Refer to *Section 11.7.8: PWR Stop 2 mode*.

- Stop 3 mode:

  The Stop 3 mode achieves the lowest power consumption while retaining some peripherals and the system configuration registers content. The SRAM content can be selected to be retained or not. All clocks in the core domain are stopped. The PLL, HSI16 and HSE32 crystal oscillators are disabled. The LSE or LSI can still run, but their digital clock must be gated off.

  The IWDG can remain active in Stop 3 mode.

  The brownout reset (BOR) always remains active and the PVD can be kept active in Stop 3 mode. The BOR0 can be configured in ultralow-power mode to further reduce power consumption.

  When exiting from Stop mode, the system clock is HSI16 at 16 MHz.

- Standby mode:

  The Standby mode is used to achieve the lowest power consumption. The internal regulator is switched off so that the core domain is powered off. The PLL1, the HSI16 RC, and the HSE32 crystal oscillators are also switched off. The LSE or LSI can still run.

  The IWDG can remain active in Standby mode.

  The BOR always remains active in Standby mode. The BOR can be configured in ultra-low-power mode to further reduce power consumption during standby mode.

  The state of each I/O during Standby mode can be selected by software: I/O with internal pull-up, internal pull-down or floating.

  When entering Standby mode, the register content is lost except for registers in the backup domain V_{DD} and Standby circuitry.

  Optionally, SRAMs can be retained in Standby with retention mode, supplied by the regulator.

  Optionally, the RTC and TAMPER can remain active when the kernel clock is LSE or LSI, supplied by the regulator (Standby with retention mode).

  Optionally, the 2.4 GHz RADIO sleep timer and configuration in RXTXRAM, and sequence SRAM can be retained in Standby with retention mode, supplied by the regulator.

  The device exits Standby mode when an NRST pin external reset, an IWDG early interrupt or reset, or a WKUP pin event occurs (configurable rising or falling edge). The device exits Standby retention mode when an RTC event (alarm, periodic wake-up,

timestamp), a TAMP tamper detection, or a 2.4 GHz RADIO sleep timer event occurs. The tamper detection can be raised either due to an external pin or due to an internal failure detection.

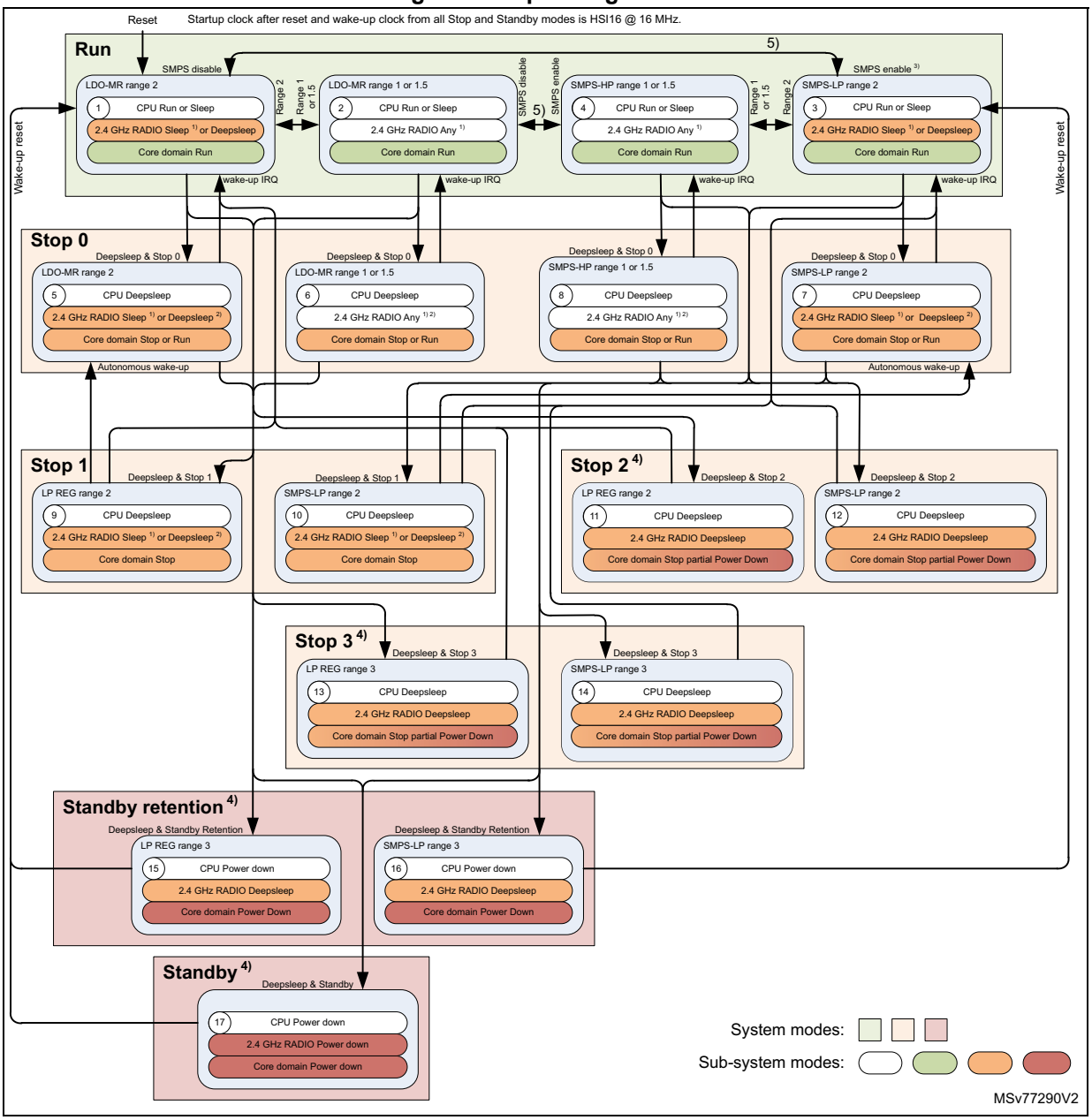
The system clock after wake-up is HSI16 16 MHz.

Refer to *Section 11.7.8: PWR Stop 2 mode*.

The operating modes and transitions are shown in *Figure 34*.

The power modes overview is shown in *Table 94*.

Figure 34. Operating modes



1. Active and Sleep states are controlled by 2.4 GHz RADIO link layer software, independent of the presence of a power supply.
2. Only in device Run mode, the 2.4 GHz RADIO operating mode can be changed by software. When the device moves to a low power mode the 2.4 GHz RADIO remains in the same mode.
3. For Standby entered from SMPS modes, wake-up is done to Run LDO range 2 mode and a subsequent move to Run SMPS range 2 mode is performed in hardware.
4. Stop 2, Stop 3, and Standby mode can only be entered when the 2.4 GHz RADIO is in Deepsleep.
5. When the 2.4 GHz radio is active, the SMPS/LDO operating mode cannot be changed. Any request to change the power operating mode is delayed and will take effect only until after the 2.4 GHz RADIO has entered Sleep mode.

Table 94. Low-power mode summary

Mode name	Entry	Wake-up source(1)	Wake-up system clock	Effect on clocks	Voltage regulator
Sleep	WFI or Return from ISR	Any interrupt	Same as before entering Sleep mode	CPU clock OFF No effect on other clocks or analog clock sources	Regulator range 1, 1.5, 2
	WFE SEVONPEDND = 0	Wake-up event			
	WFE SEVONPEDND = 1	Any interrupt, wake-up event			
Stop 0	(LPMS = Stop 0 or autonomous peripheral bus clock request active) + SLEEPDEEP bit + WFI or Return from ISR or WFE	Stop 1 wake-up sources + in Stop 0 range 1 or range 1.5: 2.4 GHz RADIO, HSECSS event, USB wake-up.	HSI16 at 16 MHz	All clocks OFF except LSE, LSI and HSI16 or HSE32 when requested as autonomous peripheral kernel or bus clock	
Stop 1	(LPMS = Stop 1 and no autonomous peripheral bus clock request) + SLEEPDEEP bit + WFI or Return from ISR or WFE	Standby with retention wake-up sources + any EXTI line. + autonomous peripherals (reduced peripheral set in Stop 2).		All clocks OFF except LSE, LSI and HSI16 when requested as autonomous peripheral kernel clock.	Regulator range 2
Stop 2	(LPMS = Stop 2 and no autonomous peripheral bus clock request and 2.4 GHz RADIO mode = Deepsleep) + SLEEPDEEP bit + WFI or Return from ISR or WFE				
	(LPMS = Stop 3 + (RADIORSB or RTCRSB = 1) and no autonomous peripheral bus clock request and 2.4 GHz RADIO mode = Deepsleep) + SLEEPDEEP bit + WFI or Return from ISR or WFE				
Stop 3	(LPMS = Stop 3 + RADIORSB and RTCRSB = 0 and no autonomous peripheral bus clock request and 2.4 GHz RADIO mode = Deepsleep) + SLEEPDEEP bit + WFI or Return from ISR or WFE	Standby wake-up sources + any EXTI line.		All clocks OFF except LSE and LSI when requested as autonomous peripheral kernel clock, their digital clock must be gated off.	Low voltage

Table 94. Low-power mode summary (continued)

Mode name	Entry	Wake-up source(1)	Wake-up system clock	Effect on clocks	Voltage regulator
Standby with retention	(LPMS = Standby + (RADIORSB or RTCRSB = 1) and no autonomous peripheral bus clock request and 2.4 GHz RADIO mode = Deepsleep) + SLEEPDEEP bit + WFI or Return from ISR or WFE	Standby wake-up sources + RTC event, TAMP event, 2.4 GHz RADIO sleep timer event.	HSI16 at 16 MHz	All clocks OFF except LSI and LSE	Regulator range 2
	(LPMS = Standby + (R[2:1]RSB n = 1 and RADIORSB and RTCRSB = 0) and no autonomous peripheral bus clock request and 2.4 GHz RADIO mode = Deepsleep) + SLEEPDEEP bit + WFI or Return from ISR or WFE	Standby wake-up sources +TAMP event.		All clocks OFF except LSE and LSI when requested as autonomous peripheral kernel clock, their digital clock is be gated off	Low voltage
Standby	(LPMS = Standby + (R[2:1]RSBn, RADIORSB and RTCRSB = 0) and no autonomous peripheral bus clock request and 2.4 GHz RADIO mode = Deepsleep) + SLEEPDEEP bit + WFI or Return from ISR or WFE	IWDG event, WKUP event, NRST external reset, BOR reset		All clocks OFF except LSI and LSE	OFF

1. Refer to *Table 95*.

Table 95. Functionalities depending on the working mode(1)

Peripheral	Run/Sleep		Stop 0			Stop 1		Stop 2		Stop 3		Standby retention		Standby	
	Range 1 or 1.5	Range 2	Range 1 or 1.5	Range 2	Wake-up capability	-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	-	Wake-up capability
CPU	A		R		-	R	-	R	-	R	-	-	-	-	-
ICACHE	A		R		-	R	-	R	-	R	-	-	-	-	-
Flash memory	O(2)		R		-	R	-	R	-	R	-	R	-	R	-
SRAM1	A		O		-	O	-	O	-	O	-	O	-	-	-
SRAM2	A		O		Y	O	-	O	-	O	-	O	-	-	-
RAMCFG	O		O		Y	R	-	R	-	R	-	-	-	-	-

Table 95. Functionalities depending on the working mode[1] (continued)

Peripheral	Run/Sleep		Stop 0			Stop 1		Stop 2		Stop 3		Standby retention		Standby	
	Range 1 or 1.5	Range 2	Range 1 or 1.5	Range 2	Wake-up capability	-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	-	Wake-up capability
SYSCFG	O		O		-	R	-	R	-	R	-	-	-	-	-
XSPI1[3]	O		R		-	R	-	-	-	-	-	-	-	-	-
DLYBXS1[3]	O		R		-	R	-	-	-	-	-	-	-	-	-
OTFDEC1[3]	O		R		-	R	-	-	-	-	-	-	-	-	-
Backup registers	A		R		-	R	-	R	-	R	-	R	-	-	-
2.4 GHz RADIO	O	R	O	R	Y	R	-	-	-	-	-	-	-	-	-
2.4 GHz RADIO SRAM	O	R	O	R	-	R	-	R	-	R	-	O	-	-	-
2.4 GHz RADIO sleep timer	O	O	O	O	Y	O	Y	O	Y	R	-	O	Y	-	-
BOR	A		A		Y	A	Y	A	Y	A	Y	A	Y	A	Y
PVD	O		O		Y	O	Y	O	Y	O	Y	-	-	-	-
HSI16 clock	O		O[4]		-	O[4]	-	O[4]	-	-	-	-	-	-	-
HSE32 clock	O		O[5]	-	-	-	-	-	-	-	-	-	-	-	-
LSI clock	O		O		-	O	-	O	-	O[6]	-	O	-	O	-
LSE clock	O		O		-	O	-	O	-	O[6]	-	O	-	O	-
CSSHSE clock security	O		O	-	Y	-	-	-	-	-	-	-	-	-	-
CSSLSE clock security	O		O		Y	O	Y	O	Y	O	Y	O	Y	-	-
RTC	O		O		Y	O	Y	O	Y	R[7]		O	Y	-	-
TAMP	O		O		Y	O	Y	O	Y	R[7]		O	Y	-	-
GPIO	O		R[8]		Y[9]	R[8]	Y[9]	R[8][10]	Y[9]	R[8][10]	Y[9]	R[11]	Y[12]	R[11]	Y[12]
IWDG	O		O		Y	O	Y	O	Y	O	Y	O	Y	O	Y
LPDMA1	O		O		Y	R	-	-	-	-	-	-	-	-	-
USART1	O		O		Y	O	Y	-	-	-	-	-	-	-	-
LPUART1	O		O		Y	O	Y	O	Y	R	-	-	-	-	-
I2C1	O		O		Y	O	Y	-	-	-	-	-	-	-	-
I2C3	O		O		Y	O	Y	O	Y	R	-	-	-	-	-
SPI3	O		O		Y	O	Y	O	Y	R	-	-	-	-	-
USB[3]	O	R	O	R	Y	R	-	-	-	-	-	-	-	-	-

Table 95. Functionalities depending on the working mode[1] (continued)

Peripheral	Run/Sleep		Stop 0			Stop 1		Stop 2		Stop 3		Standby retention		Standby	
	Range 1 or 1.5	Range 2	Range 1 or 1.5	Range 2	Wake-up capability	-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	-	Wake-up capability
ADC4	O		O		Y	O	Y	-	-	-	-	-	-	-	-
Temperature sensor	O		O[13]		Y[13]	O[13]	Y[13]	-	-	-	-	-	-	-	-
LPTIM1	O		O		Y	O	Y	O	Y	R	-	-	-	-	-
LPTIM2	O		O		Y	O	Y	-	-	-	-	-	-	-	-
PTACONV	O		R		-	R	-	-[10]	-	-[10]	-	-	-	-	-
TIMx (x = 2, 16, 17)	O		R		-	R	-	-	-	-	-	-	-	-	-
SAI1	O		R		-	R	-	-	-	-	-	-	-	-	-
RNG	O		R		-	R	-	-	-	-	-	-	-	-	-
AES	O		R		-	R	-	-	-	-	-	-	-	-	-
HASH	O		R		-	R	-	-	-	-	-	-	-	-	-
PKA	O		R		-	R	-	-	-	-	-	-	-	-	-
CRC	O		R		-	R	-	-	-	-	-	-	-	-	-
GTZC1_TZSC-(MPCWM1)[3]	O		R		-	R	-	R	-	R	-	-	-	-	-
GTZC1_TZIC	O		R		-	R	-	R	-	R	-	-	-	-	-
GTZC1_MPCBB1	O		R		-	R	-	R	-	R	-	-	-	-	-
GTZC1_MPCBB2	O		R		-	R	-	R	-	R	-	-	-	-	-
SysTick timer	O		R		-	R	-	-	-	-	-	-	-	-	-
Debug	O		O[14]		-	O[14]	-	O[14]	-	O[14]	-	O[15]	-	O[15]	-

1. A = Active, Y = yes. O = optional (can be enabled/disabled by software).R = Retained, - = not available. Gray cells highlight the wake-up capability in each mode.
2. The flash memory can be configured in power-down mode.
3. Only available on STM32WBA25xx devices.
4. Some peripherals with autonomous mode and wake-up from Stop capability can request HSI16 to be enabled. In this case, the oscillator is woken up by the peripheral, and is automatically put off when no peripheral needs it.
5. For the autonomous 2.4 GHz RADIO the HSE32 can be kept running.
6. Digital domain clock must be gated off.
7. RTC and TAMP counting needs to be reinitialized by software after Stop 3 mode exit.
8. GPIO pins from peripherals supporting autonomous mode are still operational.
9. Only GPIOs with enabled wake-up functionality in the EXTI or WKUP are able to wake up the system.
10. GPIO level retention from PTA interface in Stop 2 and Stop 3 mode can be enabled.
11. GPIO level retention in Standby modes can be enabled.

12. Only GPIOs with WKUP functionality are able to wake up the system.
13. Functional through ADC4 in autonomous mode.
14. DBGMCU remains accessible trough AP0.
15. DBGMCU remains accessible through AP0 when CDBGPWRUPREQ is set.

Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop and Standby modes while the debug features are used. This is due to the fact that the Cortex-M33 core is no longer clocked or powered.

However, by setting some configuration bits in the DBGMCU control registers, the software can be debugged even when using the low-power modes extensively. For more details, refer to *Section 41.2.5: DBG low-power modes*.

11.7.2 PWR background autonomous mode (BAM)

The devices support a background autonomous mode (BAM), that allows autonomous peripherals to be functional on its bus and kernel clock in Stop 0 mode and on its kernel clock in Stop 0 and Stop 1, and for a limited set of peripherals in Stop 2 modes (when the CPU is in Deepsleep, not running any software). Whenever an autonomous peripheral requests its bus clock, the Stop 0 mode is entered while the CPU remains in Deepsleep.

Stop 0 and Stop 1 modes

In Stop 0 and Stop 1 modes, the autonomous peripherals are ADC, LPTIM, USART, LPUART, SPI, I2C, 2.4 GHz RADIO sleep timer:

- Peripherals are autonomous only with LSE, LSI or HSI16 used as kernel clock, or HSI16 used as bus clock
- When an autonomous peripheral requests its bus clock Stop 0 mode is entered while keeping the CPU in Deepsleep.

In Stop 0 mode voltage scaling range 1 and 1.5 the 2.4 GHz RADIO can operate autonomously on HSI16 used as bus clock and HSE32 used as kernel clock. The 2.4 GHz RADIO cannot be autonomous in other Stop modes.

When requesting to enter low-power modes Stop and Standby from Run and an autonomous peripheral bus clock request is active, Stop 0 mode is entered, regardless of the low-power mode selection in LPMS bits. Entering in the LPMS selected low-power mode (Stop 1, 2, 3 and Standby) is delayed until the autonomous peripheral bus clock request is released.

When in Stop 1 mode and an autonomous peripheral requests its bus clock, the Stop 0 mode is entered.

When in Stop 0 mode and all autonomous peripheral bus clock request are deactivated, the low-power mode selected in the LPMS bits is entered. Note that Stop 2, Stop 3, and Standby modes can only be entered when the 2.4 GHz RADIO is in Deepsleep.

Note: As soon as the CPU enters Deepsleep, the system enters Stop mode and the BAM operation peripheral bus clock and SYSCLK is switched to HSI16 at 16 MHz. If an autonomous peripheral operation with higher bus clock frequency is needed, the CPU must enter Sleep and keep the system in Run with the configured Run mode SYSCLK clock frequency.

Stop 2 mode

In Stop 2 mode, the autonomous peripherals are LPTIM1, LPUART1, SPI3, I2C3:

- Peripherals are autonomous only with LSE, LSI, or HSI16 used as kernel clock.

When in Stop 2 mode an autonomous peripheral shall generate an interrupt to wake up the CPU. From Stop 2 mode, autonomous peripheral BAM operation wake up on bus clock and using the LPDMA1 is not supported.

BAM features

BAM is supported by the autonomous peripherals with the following features:

- Functionality in Stop 0, 1, and 2 mode thanks to the peripheral kernel clock request capability: the peripheral kernel clock is automatically switched on when requested by a peripheral, and automatically switched off when no peripheral requests it. For the peripheral kernel clock to be switched on, both the peripheralEN and peripheralSMEN bits in the RCC registers must be set.
- DMA transfers are supported in Stop 0 mode, thanks to the peripheral bus clock request capability: the system clock (HSI16) is automatically switched on when requested by a peripheral, and automatically switched off when no peripheral requests it. When the system clock is requested by an autonomous peripheral and both the peripheralEN and peripheralSMEN bits in the RCC registers are set, the system clock is woken up and distributed to all peripherals for which the RCC peripheralEN and peripheralSMEN bits are set. This makes possible DMA transfers between peripherals and SRAMs. The 2.4 GHz RADIO bus clock is requested independently from the RADIOEN and RADIOSMEN.
- Automatic start of the peripheral thanks to the hardware synchronous or asynchronous triggers (such as I/Os edge detection and low-power timer events)
- Wake up the CPU from Deepsleep mode through a peripheral interrupt.

The LPDMA1 is fully functional and the linked-list is updated in Stop 0 mode, allowing the different DMA transfers to be linked without any CPU intervention. This can be used to chain transfers between different peripherals, or to write peripheral registers, in order to change their configuration while in Stop 0 mode.

The DMA transfers from memory to memory can be started by hardware synchronous or asynchronous triggers, and the DMA transfers between peripherals and memories can also be gated by those triggers. In BAM mode only SRAM transfers are supported, flash memory transfers are not supported.

Here below some use-cases that can be done while remaining in Stop 0, 1, and 2 modes:

- A/D conversion triggered by a low-power timer (or any other trigger)
  - Wake up from Stop mode on analog watchdog if the A/D conversion result is out of the programmed thresholds
  - Wake up from Stop mode on DMA buffer event
- I^2C target reception or transmission, SPI reception, UART/LPUART reception
  - Wake up at the end of peripheral transfer or on DMA buffer event
- I^2C controller transfer, SPI transmission, UART/LPUART transmission, triggered by a low-power timer (or any other trigger)
  - Example: sensor periodic read
  - Wake up at the end of peripheral transfer or on DMA buffer event
- Bridges between peripherals
  - Example: A/D converted data transferred by communication peripherals
- Data transfer from/to GPIO to/from SRAM for:
  - Controlling external components
  - Implementing data transmission and reception protocols
- Data transfer from one SRAM to another one.

Here below a 2.4 GHz RADIO use-case that can be done while remaining in Stop 0 mode voltage scaling range 1 or range 1.5:

- Transmit and receive data packets.
  - Wake up at the end of the scheduled event.

11.7.3 PWR Run mode

Slowing down system clocks

In Run mode, the speed of the system clocks (SYSCLK, hclk, pclk) can be reduced by programming the prescaler registers. These prescalers can also be used to slow down the peripherals before entering the Sleep mode.

For more details, refer to *Section 12: Reset and clock control (RCC)*.

In addition to slowing down the system clocks the voltage scaling range can be changed. When the system clock frequency is low enough, range 2 can be entered.

Peripheral clock gating

In Run mode, the hclk and pclk for individual peripherals and memories can be stopped at any time to reduce the power consumption. The peripheral clock gating is controlled by the RCC_AHBxENR and RCC_APBxENR registers.

In Sleep and Stop 0, 1, and 2 modes, the hclk and pclk for individual peripherals and memories can be stopped automatically. The Sleep and Stop 0, 1, and 2 mode peripheral clock gating is controlled by the RCC_AHBxSMENR and RCC_APBxSMENR registers. For autonomous peripherals to be able to request their clocks the peripheral bus clock in Sleep and Stop 0, 1, and 2 mode must be enabled. Except for the 2.4 GHz RADIO, which requests its bus clock independently from the settings in the RADIOEN and RADIOSMEN register bits.

For the 2.4 GHz RADIO the bus clock is running only in Sleep and Stop 0, 1, and 2 modes when the STRADIOCLKON is set, or when the 2.4 GHz RADIO is active and RADIOEN and RADIOSMEN are set.

11.7.4 PWR low-power modes

Entering into a low-power mode

The MCU enters in low-power modes by

- The CPU executing the WFI (wait for interrupt), or WFE (wait for event) instructions, or when the SLEEPONEXIT bit in the Cortex-M33 system control register is set on Return from ISR.

Entering into a low-power mode through WFI or WFE is executed only if no interrupt is pending or no event is pending.

Software must enter low-power Stop and Standby modes only when voltage scaling is ready (ACTVOSRDY = 1).

When an autonomous peripheral bus clock request is active only Stop 0 is entered, regardless of the low-power mode set in LPMS register bits.

Stop 2, Stop 3, and Standby modes are only entered when the 2.4 GHz RADIO is in Deepsleep mode.

Exiting a low-power mode

The way the CPU exits the Sleep or Stop mode depends on the way the low-power mode was entered:

- If the WFI instruction or *Return from ISR* is used to enter the low-power mode, any peripheral interrupt acknowledged by the NVIC can wake up the device.
- If the WFE instruction is used to enter the low-power mode, the CPU exits the low-power mode as soon as an event occurs. The wake-up event can be generated by:
  - An NVIC IRQ interrupt:
    When SEVONPEND = 0 in the Cortex-M33 system control register
    By enabling an interrupt in the peripheral control register and in the NVIC. When the CPU resumes from WFE, the peripheral interrupt pending bit and the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) must be cleared. Only NVIC interrupts with high enough priority wake up and interrupt the CPU.
    When SEVONPEND = 1 in the Cortex-M33 system control register
    By enabling an interrupt in the peripheral control register and optionally in the NVIC. When the CPU resumes from WFE, the peripheral interrupt pending bit and when enabled the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) must be cleared. All NVIC interrupts wake up the MCU, even the disabled ones. Only enabled NVIC interrupts with high enough priority wake up and interrupt the CPU.
  - An event:
    Configuring a EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the EXTI peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bits corresponding to the event line is not set. It can be necessary to clear the interrupt flag in the peripheral.

The CPU exits Standby mode through a reset. After waking up from Standby mode, the

program execution restarts in the same way as after a reset (boot pin sampling, option bytes loading, reset vector is fetched).

Caution: When the device is in Stop mode, a peripheral interrupt powers on an internal oscillator. The corresponding NVIC interrupt channel must be enabled to allow the interrupt to wake up the CPU from Stop mode. It is not allowed to disable a peripheral interrupt by disabling only the NVIC channel while keeping the peripheral interrupt enabled, as the device could remain in Stop 0, 1, and 2 mode with clock ON.
The peripherals with autonomous mode feature are able to generate an AHB or APB clock request when the device is in Stop 0, 1, and 2 mode, depending on their internal events. The software must ensure that either DMA transfer or interrupt is served, by configuring properly and in a consistent way the RCC, the autonomous peripherals, the DMA channels, and NVIC. Note that when an autonomous peripheral requests the bus clock in Stop mode, the AHB and APB clocks are distributed to all enabled peripherals on the same AHB or APB bus. Consequently, enabled peripherals, even without autonomous mode capability, are temporarily clocked and can also generate an interrupt during this time. These peripheral interrupts also wake up the device from Stop mode.

11.7.5 PWR Sleep mode

I/O states in Sleep mode

In Sleep mode, all I/O pins keep the same state as in Run mode. In addition I/O pins can be toggled through DMA or other active communication peripherals.

To further reduce the power consumption in Sleep mode, disabling the peripheral clocks in Sleep mode can be performed automatically by resetting the corresponding bit in the RCC_AHBxSMENR and RCC_APBxSMENR registers.

Entering the Sleep mode

The MCU enters the Sleep mode as described in *Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is cleared (see the table below for details on how to enter the Sleep mode).

Exiting the Sleep mode

The MCU exits the Sleep mode as described in *Exiting a low-power mode* (see the table below for details on how to exit the Sleep mode).

Table 96. Sleep mode

Sleep mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is cleared in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending
	On return from ISR while: – SLEEPDEEP bit is cleared in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending

Table 96. Sleep mode (continued)

Sleep mode	Description
Mode exit	If WFI or Return from ISR was used for entry Interrupt (see *Table 140: Vector table*) If WFE was used for entry and SEVONPEND = 0: Wake-up event (see *Section 18: Extended interrupts and event controller (EXTI)*) If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC (see *Table 140: Vector table*) or wake-up event (see *Section 18: Extended interrupts and event controller (EXTI)*)
Wake-up latency	None

11.7.6 PWR Stop 0 mode

The Stop mode is based on the Core Deepsleep mode combined with peripheral clock gating. The voltage regulator configuration, as used in Run mode, remains the same in Stop 0 mode. In Stop mode, all clocks in the core domain are stopped. The PLL1, HSI16, and HSE32 oscillators are disabled.

Some peripherals with the BAM capability can switch on HSI16 or HSE32 for transferring data (see *Section 11.7.2* for details). BAM capable peripherals using HSE32 require the use of Stop 0 mode voltage scaling range 1 or range 1.5.

To reduce the power consumption in Stop mode, disabling the peripheral clocks can be performed automatically by resetting the corresponding bit in the RCC_AHBxSMENR and RCC_APBxSMENR registers. This bit must be set for the autonomous peripherals requesting clocks in Stop 0, 1, and 2 mode.

All register contents are preserved, SRAMs can fully or partially be retained to further reduce consumption.

I/O states in Stop 0 mode

In the Stop 0 mode, all I/O pins keep the same state as in the Run mode. In addition I/O pins can be toggled through DMA or other BAM communication peripherals.

Entering the Stop 0 mode

The MCU enters the Stop 0 mode as described in *Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is set. The regulator is kept in the same regulation mode and voltage scaling range as used in Run mode (see the table below for details on how to enter the Stop 0 mode).

If the flash memory programming is ongoing, the Stop 0 mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, the Stop 0 mode entry is delayed until the APB access is finished.

In Stop 0 mode, the following features can be selected by programming the individual control bits:

- The independent watchdog (IWDG) is started by writing to its key register or by hardware option. Once started, it cannot be stopped except by a reset (see *Section 32.4: IWDG functional description*).
- The real-time clock (RTC) and tamper (TAMP) is configured by the RTCSEL bit in the *RCC backup domain control register (RCC_BDCR1)*.
- The internal RC oscillator LSI clock, or LSI clock divided by 128, is configured by the LSI1ON and LSI1PREDIV bits in RCC_BDCR1.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_BDCR1.
- The 2.4 GHz RADIO sleep timer is configured through the link layer software.
- The SRAM1 retention is configured by the SRAM1PDS bit in *PWR control register 2 (PWR_CR2)*.
- The SRAM2 retention is configured by the SRAM2PDSx bit in *PWR control register 2 (PWR_CR2)*.
- The peripheral SRAM retention is configured by PKARAMPDS and PRAMPDS in *PWR control register 2 (PWR_CR2)*.

In Stop 0 mode voltage scaling range 1 and 1.5, the following features can be selected by programming the individual control bits:

- In addition to the Stop 0 features above, the 2.4 GHz RADIO configured through the link layer software.

Several peripherals can be autonomous in Stop 0 mode, increasing power consumption if enabled (see *Section 11.7.2* for more details).

The PVD can be used in Stop mode. If not needed, it must be disabled by software to reduce power consumption.

The ADC4 and the temperature sensor can consume power during the Stop mode, unless they are disabled before entering this mode.

Entering the Stop 1, 2, 3 and Standby modes from Stop 0

When entering low power with Stop 1, 2, 3 or Standby selected in LPMS and an autonomous peripheral bus clock request is active, Stop 0 mode is entered instead. Only when all autonomous peripheral bus clock requests are deasserted the LPMS selected low-power mode is subsequently entered. Stop 2, Stop 3, and Standby modes are only entered when the 2.4 GHz RADIO is in Deepsleep.

Exiting the Stop 0 mode

The MCU exits the Stop 0 mode as described in *Exiting a low-power mode* (see *Table 97*).

When exiting Stop 0 mode by issuing an interrupt or a wake-up event, HSI16 is selected as system clock.

Several peripherals are autonomous in Stop mode, and can generate interrupts with wake-up from Stop capability. All peripheral clocks must be enabled to allow a wake-up from Stop interrupt (see *Peripheral clock gating*).

When exiting the Stop 0 mode, the MCU is in Run mode with the same voltage scaling range as before entering Stop 0 mode.

The MCU can enter another low-power mode, as selected in LPMS, from the Stop 0 mode when all autonomous peripheral bus clocks requests are deasserted.

Table 97. Stop 0 mode

Stop 0 mode	Description
Mode entry from Run	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending
	– LPMS = Stop 0 in PWR_CR1 or autonomous peripheral bus clock request is active
	Note: To enter Stop 0 mode, all EXTI line pending bits (in the EXTI rising edge pending register (EXTI_RPR1) and EXTI falling edge pending register (EXTI_FPR1)), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop 0 mode entry procedure is ignored and the program execution continues.
Mode entry from Stop 1	Autonomous peripheral bus clock request active

Table 97. Stop 0 mode (continued)

Stop 0 mode	Description
Mode exit to Run	If WFI or Return from ISR was used for entry: – Any EXTI line configured in interrupt mode (the corresponding EXTI interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (see *Table 140: Vector table*). – 2.4 GHz RADIO sleep timer event, WKUPx event, RTC event, TAMP event, IWDG event, NRST external reset, BOR reset. – Any peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request (the peripheral vector must be enabled in the NVIC). – In addition only from Stop 0 voltage scaling range 1 and range 1.5, 2.4 GHz RADIO interrupt and HSECSS interrupt. If WFE was used for entry and SEVONPEND = 0: – Any EXTI line configured in event mode (see *Section 18: Extended interrupts and event controller (EXTI)*). If WFE was used for entry and SEVONPEND = 1: – Any EXTI line configured in interrupt mode (even if the corresponding EXTI interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (see *Table 140: Vector table*). – Any EXTI line is configured in event mode (see *Section 18: Extended interrupts and event controller (EXTI)*). – 2.4 GHz RADIO sleep timer event, WKUPx event, RTC event, TAMP event, IWDG event, NRST external reset, BOR reset. – Any peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request (the peripheral vector must be enabled in the NVIC). – In addition only from Stop 0 voltage scaling range 1 and range 1.5, 2.4 GHz RADIO interrupt and HSECSS interrupt.
	Note: All peripheral clocks must be enabled to allow this peripheral to generate a wake-up from Stop interrupt (peripheralEN, and peripheralSMEN bits must be set in the RCC, and a functional independent clock must be selected).
Mode exit to low-power modes	All autonomous peripheral bus clock requests deasserted.
Wake-up latency	HSI16 wake-up time when applicable and flash wake-up time from Stop 0 mode.

11.7.7 PWR Stop 1 mode

The Stop 1 mode is similar to the Stop 0 mode in voltage scaling range 2, except that the regulator is in low-power mode (see the table below for details on how to enter and exit Stop 1 mode).

The BOR is always available in Stop 1 mode. When not using BAM operation with HSI16 as kernel clock, the BOR0 can be forced in ultralow-power mode by the ULPMEN bit in the PWR_CR1 to reach the lowest power consumption.

Entering the Stop 1mode

The MCU enters the Stop 1 mode in the same way as entering Stop 0. Whenever an autonomous peripheral bus clock request is active, Stop 1 mode is not entered and Stop 0 mode is entered instead.

Entering the Stop 0 mode from Stop 1

When in low-power Stop 1 mode, an autonomous peripheral bus clock request is activated. The Stop 0 mode voltage scaling range 2 is entered with HSI16 as system clock and HDIV5 is set to divide-by-2 by hardware.

Exiting the Stop 1 mode

The MCU exits the Stop 1 mode as described in *Exiting a low-power mode* (see *Table 98*).

When exiting Stop 1 mode by issuing an interrupt or a wake-up event, Run voltage scaling range 2 with HSI16 is selected as the system clock, and HDIV5 is set to divide-by-2 by hardware. Before entering Stop 1 mode, the software must configure adequate FLASH wait states latency to at least 1 in FLASH_ACR register and SRAMs wait states to at least 1 in RAMCFG_MxCR.

Table 98. Stop 1 mode

Stop 1 mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending
	– LPMS = Stop 1 in PWR_CR1 and all autonomous peripheral bus clock requests deasserted. – LPMS = Stop 2 in PWR_CR1 and 2.4 GHz RADIO not in Deepsleep. – LPMS = Stop 3 in PWR_CR1 and 2.4 GHz RADIO not in Deepsleep. – LPMS = Standby in PWR_CR1 and all autonomous peripheral bus clock requests deasserted and 2.4 GHz RADIO not in Deepsleep.
	Note: To enter Stop 1 mode, all EXTI line pending bits (in EXTI rising edge pending register (EXTI_RPR1) and EXTI falling edge pending register (EXTI_FPR1)), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop 1 mode entry procedure is ignored and the program execution continues.

Table 98. Stop 1 mode (continued)

Stop 1 mode	Description
Mode exit to Run	If WFI or Return from ISR was used for entry – Any EXTI line configured in interrupt mode (the corresponding EXTI interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (see *Table 140: Vector table*). – 2.4 GHz RADIO sleep timer event, WKUPx event, RTC event, TAMP event, IWDG event, NRST external reset, BOR reset – Any peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request (the peripheral vector must be enabled in the NVIC) If WFE was used for entry and SEVONPEND = 0: – Any EXTI line configured in event mode (see *Section 18: Extended interrupts and event controller (EXTI)*) If WFE was used for entry and SEVONPEND = 1: – Any EXTI line configured in interrupt mode (even if the corresponding EXTI interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (see *Table 140: Vector table*). – Any EXTI line is configured in event mode (see *Section 18: Extended interrupts and event controller (EXTI)*)) – 2.4 GHz RADIO sleep timer event, WKUPx event, RTC event, TAMP event, IWDG event, NRST external reset, BOR reset Any peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request (the peripheral vector must be enabled in the NVIC)
	Note: All peripheral clocks must be enabled to allow this peripheral to generate a wake-up from Stop interrupt (peripheralEN and peripheralSMEN bits must be set in the RCC, and a functional independent clock must be selected).
Mode exit to Stop 0	Autonomous peripheral bus clock request active.
Wake-up latency	HSI16 wake-up time and regulator wake-up time from low-power mode + flash wake-up time from Stop 1 mode.

11.7.8 PWR Stop 2 mode

The Stop 2 mode is similar to the Stop 1 mode except that part of the peripherals is powered down (see the *Table 95: Functionalities depending on the working mode* for details).

The BOR is always available in Stop 2 mode. When not using autonomous peripheral operation with HSI16 as kernel clock, the BOR0 can be forced in ultralow-power mode by the ULPMEN bit in the PWR_CR1 to reach the lowest power consumption.

I/O states in Stop 2 mode

Before entering Stop 2 mode the peripherals that are not retained shall be disabled and unselected from the GPIO alternate functions. Except for the PTACONV, where the GPIO state can be retained when enabled in the PTASREN register bit, the GPIO retains the output level. When entering Stop 2 mode, PTACONV signals enabled for Stop 2 mode retention keep their state during and after exiting from Stop 2 mode until the PTASR bit is cleared by software. When entering Stop 2 mode the PTASR bit is set by hardware when enabled in PTASREN. The PTASR only controls output levels, it does not affect the GPIO

pulls, analog, input and output states. When exiting from Stop 2 and before removing the PTACONV retention, the PTACONV peripheral must be reinitialized. Once the PTACONV peripheral configuration is reconfigured, disable the retention in PTASR.

Entering the Stop 2 mode

The MCU enters the Stop 2 mode in the same way as entering Stop 0. Whenever an autonomous peripheral bus clock request is active, Stop 2 mode is not entered and Stop 0 mode is entered instead.

Stop 2 mode is only entered when the 2.4 GHz RADIO is in Deepsleep mode. Whenever the 2.4 GHz RADIO is in Active or Sleep mode, Stop 0 or Stop 1 mode is entered instead.

Note: Autonomous peripherals in Stop 2 mode should be configured by software to only request a bus clock when also generating a peripheral IRQ event to wake up the CPU.

Exiting the Stop 2 mode

The MCU exits the Stop 2 mode as described in *Exiting a low-power mode* (see *Table 98*). The STOP2F status flag in the *PWR status register (PWR_SR)* indicates that the MCU was in Stop 2 mode. The powered down peripherals registers are reset after wake-up from Stop 2.

When exiting Stop 2 mode by issuing an interrupt or a wake-up event, Run range 2 with HSI16 is selected as the system clock and HDIV5 is set to divide-by-2 by hardware. Before entering Stop 2 mode software must configure adequate FLASH wait states latency to at least 1 in FLASH_ACR register and SRAMs wait states to at least 1 in RAMCFG_MxCR.

Table 99. Stop 2 mode

Stop 2 mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending
	– LPMS = Stop 2 in PWR_CR1 and all autonomous peripheral bus clock requests deasserted and 2.4 GHz RADIO in Deepsleep. – LPMS = Stop 3 in PWR_CR1, RADIORSB or RTCRSB = 1, and all autonomous peripheral bus clock requests de-asserted and 2.4 GHz RADIO in Deepsleep.
	Note: To enter Stop 2 mode, all EXTI line pending bits (in EXTI rising edge pending register (EXTI_RPR1) and EXTI falling edge pending register (EXTI_FPR1)), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop 2 mode entry procedure is ignored and the program execution continues.

Table 99. Stop 2 mode (continued)

Stop 2 mode	Description
Mode exit to Run	If WFI or Return from ISR was used for entry – Any EXTI line configured in interrupt mode (the corresponding EXTI interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (see *Table 140: Vector table*). – 2.4 GHz RADIO sleep timer event, WKUPx event, RTC event, TAMP event, IWDG event, NRST external reset, BOR reset – Stop 2 mode retained peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request (the peripheral vector must be enabled in the NVIC) If WFE was used for entry and SEVONPEND = 0: – Any EXTI line configured in event mode (see *Section 18.3: EXTI functional description*) If WFE was used for entry and SEVONPEND = 1: – Any EXTI line configured in interrupt mode (even if the corresponding EXTI interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (see *Table 140: Vector table*). – Any EXTI line is configured in event mode (see *Section 18.3: EXTI functional description*)) – 2.4 GHz RADIO sleep timer event, WKUPx event, RTC event, TAMP event, IWDG event, NRST external reset, BOR reset – Stop 2 mode retained peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request (the peripheral vector must be enabled in the NVIC)
	Note: All peripheral clocks must be enabled to allow this peripheral to generate a wake-up from Stop interrupt (peripheralEN and peripheralSMEN bits must be set in the RCC, and a functional independent clock must be selected).
Wake-up latency	HSI16 wake-up time and regulator wake-up time from low-power mode + flash wake-up time from Stop 2 mode.

11.7.9 PWR Stop 3 mode

The Stop 3 mode is similar to the Stop 2 mode except that there is no wake-up from autonomous peripherals (see the *Table 95: Functionalities depending on the working mode* for details).

The BOR is always available in Stop 3 mode. The BOR0 can be forced in ultralow-power mode by the ULPMEN bit in the PWR_CR1 to reach the lowest power consumption.

I/O states in Stop 3 mode

Before entering Stop 3 mode the peripherals that are not retained must be disabled and unselected from the GPIO alternate functions. Except for the PTACONV, where the GPIO state can be retained when enabled in the PTASREN register bit, the GPIO retains the output level. When entering Stop 3 mode, PTACONV signals enabled for Stop 3 mode retention keep their state during and after exiting from Stop 3 mode until the PTASR bit is cleared by software. When entering Stop 3 mode the PTASR bit is set by hardware when enabled in PTASREN. The PTASR only controls output levels, it does not affect the GPIO pulls, analog, input and output states. When exiting from Stop 3 and before removing the

PTACONV retention, the PTACONV peripheral must be reinitialized. Once the PTACONV peripheral configuration is reconfigured, disable the retention in PTASR.

Entering the Stop 3 mode

The MCU enters the Stop 3 mode in the same way as entering Stop 0. Whenever an autonomous peripheral bus clock request is active, Stop 3 mode is not entered and Stop 0 mode is entered instead.

When RTCRSB or RADIORSB is 1, Stop 3 mode is not entered and Stop 2 mode is entered instead.

Regardless of an active autonomous peripheral kernel clock request, when entering Stop 3 mode the HSI16 oscillator and all clocks in the V_{CORE} domain stopped. LSI and LSE clocks in the digital domain shall be gated off by software in the RCC, by LSIGATEEN and LSEGATEEN.

Stop 3 mode is only entered when the 2.4 GHz RADIO is in Deepsleep mode. Whenever the 2.4 GHz RADIO is in Active or Sleep mode, Stop 0 or Stop 1 mode is entered instead.

Exiting the Stop 3 mode

The MCU exits the Stop 3 mode as described in *Exiting a low-power mode* (see *Table 98*). The STOP2F status flag in the *PWR status register (PWR_SR)* indicates that the MCU was in Stop 3 mode. The powered down peripherals registers are reset after wake-up from Stop 3.

When exiting Stop 3 mode by issuing an interrupt or a wake-up event, Run voltage scaling range 2 with HSI16 is selected as the system clock and HDIV5 is set to divide-by-2 by hardware. Before entering Stop 3 mode, the software must configure adequate FLASH wait states latency to at least 1 in FLASH_ACR register and SRAMs wait states to at least 1 in RAMCFG_MxCR. The gating of LSI and LSE clocks in the digital domain shall be removed by software in the RCC LSIGATEEN and LSEGATEEN. The RTC and TAMP counting need to be reinitialized by software.

Table 100. Stop 3 mode

Stop 3 mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending
	– LPMS = Stop 3 in PWR_CR1, RTCRSB and RADIORSB are 0, and all autonomous peripheral bus clock requests deasserted and 2.4 GHz RADIO in Deepsleep.
	Note: To enter Stop 3 mode, all EXTI line pending bits (in EXTI rising edge pending register (EXTI_RPR1) and EXTI falling edge pending register (EXTI_FPR1)), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop 3 mode entry procedure is ignored and the program execution continues.

Table 100. Stop 3 mode (continued)

Stop 3 mode	Description
Mode exit to Run	If WFI or Return from ISR was used for entry – Any EXTI line configured in interrupt mode (the corresponding EXTI interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (see *Table 140: Vector table*). – WKUPx event, IWDG event, NRST external reset, BOR reset If WFE was used for entry and SEVONPEND = 0: – Any EXTI line configured in event mode (see *Section 18.3: EXTI functional description*) If WFE was used for entry and SEVONPEND = 1: – Any EXTI line configured in interrupt mode (even if the corresponding EXTI interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (see *Table 140: Vector table*). – Any EXTI line is configured in event mode (see *Section 18.3: EXTI functional description*)) – WKUPx event, IWDG event, NRST external reset, BOR reset
Wake-up latency	HSI16 wake-up time and regulator wake-up time from low-power mode + flash wake-up time from Stop 3 mode.

11.7.10 PWR Standby mode

It is based on the Cortex-M33 Deepsleep mode, with the voltage regulators disabled (except when SRAM1, SRAM2, 2.4 GHz RADIO RAMs and 2.4 GHz RADIO sleep timer are retained). The PLL1, HSI16, and HSE32 oscillators are also switched off.

The register contents are lost except for SRAMs and registers in the retention domain and backup domain V_{CORE} when enabled and the backup domain V_{DD} and Standby circuitry (see *Figure 30*). SRAMs' content can be preserved depending on the RxRSBy bits configuration in PWR_CR1. In this case, the regulator is ON and provides the supply to SRAMs.

Also the 2.4 GHz RADIO SRAMs can be retained and the 2.4 GHz RADIO sleep timer kept operational depending on RADIORSB bit configuration in PWR_CR1.

The RTC and TAMP can be retained when enabled in RTCRSB bit in PWR_RC1.

The BOR is always available in Standby mode. The ULPMEN bit in the PWR_CR1 register must be configured to 1 to reach the lowest power consumption by forcing the BOR0 in ultralow-power mode.

I/O states in Standby mode

In the Standby mode, the GPIOs are by default in floating state. If Standby GPIO retention is enabled in the PWR_IORETENRx register, the GPIO retains the pull or output level. When entering Standby mode, GPIOs that are enabled for Standby mode retention keep their pull or level during and after exiting from Standby mode until the PWR_IORETRx bit is cleared by software. When entering Standby mode the PWR_IORETRx bit is set by hardware for the

GPIOs with Standby retention enabled. Only once PWR_IORETRx is cleared by software, the GPIO configuration is switched to that defined below:

- When GPIO is not selecting an alternate function GPIO configuration is applied
- When GPIO is selecting an alternate function the selected peripheral alternate function control is applied

The GPIO Standby retention enable information in PWR_IORETENRx and PWR_IORETRx are retained in Standby mode.

Note: The Standby GPIO retention level cannot be guaranteed when the GPIO port pin is connected to a low impedance destination.

Table 101. GPIO retention pin with pull-up and pull-down

PWR_IORETEN	GPIO port pin configuration before entering Standby	GPIO port pin configuration in Standby
0	Any	High-Z
1	Input no pulls	High-Z
	Input or output with pull-up	Pull-up
	Input or output with pull-down	Pull-down
	Output no pulls driving level high	Pull-up
	Output no pulls driving level low	Pull-down

For GPIO port pins enabled to be functional in Standby modes, the Standby GPIO retention can also be controlled in the PWR_IORETENRx register. The following GPIO functions are available in Standby modes:

- RTC outputs on PC13 and/or PB2 (standby GPIO retention must be disabled), when retention is enabled with register bit RTCRSB.
- TAMP tamper pins, when retention enabled with register bit RTCRSB.
- WKUPx_y wake-up pins
- LSE pins on PC14 and/or PC15 (Standby GPIO retention must be disabled)
- JTDO/TRACESWO on PB3 when debugger is connected and CDBGPWRUPREQ is set (Standby GPIO retention must be disabled)

When waking up from Standby with a system reset (NRST) GPIO retention is removed before the software can reconfigure the GPIO port pins in the GPIO peripheral.

Entering Standby mode

The MCU enters the Standby mode as described in *Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is set (see *Table 102*).

Whenever an autonomous peripheral bus clock request is active, Standby mode is not entered and Stop 0 mode is entered instead.

Standby mode is only entered when the 2.4 GHz RADIO is in Deepsleep mode. Whenever the 2.4 GHz RADIO is in Active or Sleep mode Stop mode is entered instead.

In Standby mode, the following features can be selected by programming individual control bits:

- The independent watchdog (IWDG) is started by writing to its key register or by hardware option. Once started it cannot be stopped except by a reset (see *Section 32.4: IWDG functional description*).
- The internal RC oscillator LSI clock or LSI clock divided by 128 is configured by the LSI1ON and LSI1PREDIV bits in RCC_BDCR1.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_BDCR1.
- The real-time clock (RTC) and tamper (TAMP) is retained by the RTCRSB and is configured by the RTCSEL bit in *RCC backup domain control register (RCC_BDCR1)*. Available in Standby retention mode.
- The 2.4 GHz RADIO SRAMs and 2.4 GHz RADIO sleep timer retained by the RADIORSB bit in *PWR control register 1 (PWR_CR1)*. Available in Standby retention mode.
- SRAM1 retention configured by the R1RSB bit in *PWR control register 1 (PWR_CR1)*. Available in Standby retention mode.
- SRAM2 retention configured by the R2RSBx bit in *PWR control register 1 (PWR_CR1)*. Available in Standby retention mode.

When RTCRSB or RADIORSB is 1, voltage range 2 is used in Standby retention mode. When RTCRSB and RADIORSB are 0 and at least one RxRSBy is 1, low voltage is used in Standby retention mode.

Exiting Standby mode

The MCU exits the Standby mode as described in *Exiting a low-power mode*. The SBF status flag in the *PWR status register (PWR_SR)* indicates that the MCU was in Standby mode *(see Table 102* for more details on how to exit Standby mode).

Table 102. Standby mode

Standby mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending
	– LPMS = Standby in PWR_CR1 and all autonomous peripheral bus clock requests deasserted and the 2.4 GHz RADIO in Deepsleep. – WUFx bits cleared in PWR_WUSR – RTC and TAMP flags corresponding to the chosen wake-up source cleared – 2.4 GHz RADIO sleep timer wake-up source cleared.
	In Standby retention mode the Vcore level depend on RTCRSB and RADIORSB. Refer to *Table 94: Low-power mode summary*.
	Note: To enter Standby mode, all EXTI line pending bits (in EXTI rising edge pending register (EXTI_RPR1) and EXTI falling edge pending register (EXTI_FPR1)), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Standby mode entry procedure is ignored and the program execution continues.
Mode exit	WKUPx event, IWDG event, NRST external reset, BOR reset In addition only from Standby retention RTC event, TAMP event, 2.4 GHz RADIO sleep timer event.
Wake-up latency	Reset phase

11.7.11 Power modes output pins

In order to help the debug, three signals are available as device pins alternate functions:

- **SLEEP**

  When set, SLEEP indicates that the CPU is in Sleep mode:

  – WFI or WFE has been executed, and CPU stops execution CPU hclk1 stopped.

  When cleared, SLEEP indicates that the CPU is in Run mode.

- **STOP**

  When set, STOP indicates that the device is in Stop mode, meaning that the following conditions are true:

  – WFI or WFE has been executed with CPU SLEEPDEEP = 1.

  When cleared, STOP indicates that the device is in Run or Sleep mode with the CPU clocked.

Note: After WFI or WFE has been executed, the SYSCLK clock is kept running in Stop 0 mode when an autonomous peripheral requests its bus clock. The peripherals bus clock request can prevent the device to enter the selected low-power mode, which enters Stop 0 mode instead (refer to Section 11.7.2: PWR background autonomous mode (BAM) and Section 11.7.4: PWR low-power modes).

Table 103 explains the MCU power mode depending on these signals states.

Table 103. Power modes output states versus MCU power modes

SLEEP	STOP	MCU power modes(1)
0	0	Run mode (CPU executing)
1	0	Sleep mode (CPU Sleep)
X	1	Stop mode (CPU Deepsleep)

1. SLEEP and STOP are generated in core domain, consequently they are not driven in Standby mode.

11.8 PWR security and privileged protection

11.8.1 PWR security protection

TrustZone security is activated by the TZEN user option bit in the FLASH_OPTR. Some PWR register fields can be secured against nonsecure access.

The PWR TrustZone security allows the following features to be secured through the PWR_SECCFGR register:

- Low-power mode
- Wake-up (WKUP) pins
- Voltage detection
- Backup domain control

Other PWR configuration bits are secure when:

- The system clock selection is secure in RCC: the voltage scaling (VOS) configuration is secure.
- The I/O Standby mode retention configuration is secure when the corresponding GPIO is secure.

If SPRIV is set in the *PWR privilege control register (PWR_PRIVCFGR)*, the PWR_SECCFGR register can be written only by secure and privileged access. If SPRIV is cleared, PWR_SECCFGR can be written only by secure access, privileged, or unprivileged.

PWR_SECCFGR can be read by secure, nonsecure, privileged and unprivileged access.

A nonsecure write access to PWR_SECCFGR is WI and generates an illegal access event and an interrupt if enabled in the GTZC.

When the TrustZone security is disabled (TZEN = 0), PWR_SECCFGR is RAZ/WI and all other registers are nonsecure.

When a peripheral is configured as secure, its related PWR feature control bits are also secure in the associated registers. PWR_PUCRx, PWR_PDCRx, PWR_RADIOSCR, and PWR_VOSR.

A peripheral is secure when:

- For securable peripherals by GTZC-TZSC (TrustZone security controller), by the SEC security bit in the secure configuration registers corresponding to this peripheral.
- For TrustZone-aware peripherals, a security feature of this peripheral is enabled through its dedicated bits.

Table 104 gives a summary of the PWR secured bits following the security configuration bit in PWR_SECCFGR.

A nonsecure access to a secure-protected register bit is denied:

- The secured bits are not written (WI) with a nonsecure write access.
- The secured bits are read as 0 (RAZ) with a nonsecure read access.

Table 104. PWR Security configuration summary

Secure configuration register	Security configuration bit	Register name	Secured bits	Nonsecure access on secure bits
PWR_SECCFGR	Not applicable(1)	PWR_SECCFGR	All bits	Read OK. WI and illegal access event
PWR_SECCFGR	Not applicable(2)	PWR_PRIVCFGR	SPRIV	Read OK. WI
PWR_SECCFGR	LPMSEC	PWR_CR1	All bits	RAZ/WI
		PWR_CR2	All bits	
		PWR_SR	CSSF	WI
PWR_SECCFGR	VDMSEC	PWR_CR3	All bits	RAZ/WI
		PWR_SVMCR	All bits	RAZ/WI
PWR_SECCFGR	VBSEC	PWR_DBPR	All bits	RAZ/WI
PWR_SECCFGR	WUPxSEC (x = 1 to 8)	PWR_WUCR1	WUPENx	RAZ/WI
		PWR_WUCR2	WUPPx	RAZ/WI
		PWR_WUCR3	WUSELx	RAZ/WI
		PWR_WUSCR	CWUFx	WI
GTZC1_TZSC_ SECCFGR	RADIOSEC	PWR_RADIOSCR	REGPABYPEN	RAZ/WI
			REGPASEL	RAZ/WI
RCC_SECCFGR	SYSCLKSEC	PWR_VOSR	VOS	RAZ/WI
GPIOx_SECCFGR (x = A to C and H)	SECy (y = 0 to 15)	PWR_IORETENRx	ENy	RAZ/WI
		PWR_IORETRx	RETy	RAZ/WI
GTZC_TZSC_ SECCFGR	PTACONVSEC	PWR_S2RETR	PTASREN	RAZ/WI
			PTASR	RAZ/WI

1. PWR_SECCFGR is always secure.
2. PWR_PRIVCFGR.SPRIV is always secure.

11.8.2 PWR privileged protection

By default, after a reset, all PWR registers can be read or written with both privileged and unprivileged accesses, except PWR_PRIVCFGR that can be written with privileged access only. PWR_PRIVCFGR can be read by secure and nonsecure, privileged, and unprivileged accesses.

The SPRIV bit in PWR_PRIVCFGR can be written with secure privileged access only. This bit configures the privileged access of all PWR secure functions (defined by PWR_SECCFGR, GTZC, RCC, or GPIO as shown in *Table 104*).

When the SPRIV bit is set in PWR_PRIVCFGR:

- The PWR secure bits can be written only with privileged access, including PWR_SECCFGR.
- The PWR secure bits can be read only with privileged access except PWR_SECCFGR and PWR_PRIVCFGR that can be read by privileged or unprivileged access.
- An unprivileged access to a privileged PWR bit or register is discarded: the bits are read as zero and the write to these bits is ignored (RAZ/WI).

The NSPRIV bit of PWR_PRIVCFGR can be written with privileged access only, secure or nonsecure. This bit configures the privileged access of all PWR securable functions that are configured as nonsecure (defined by PWR_SECCFGR, GTZC, RCC, or GPIO as shown in *Table 104*).

When the NSPRIV bit is set in PWR_PRIVCFGR:

- The PWR securable bits that are configured as nonsecure, can be written only with privileged access.
- The PWR securable bits that are configured as nonsecure, can be read only with privileged access except PWR_PRIVCFGR that can be read by privileged or unprivileged accesses.
- The VOSRDY bit in PWR_VOSR and the registers PWR_SR, PWR_SVMSR, PWR_BDSR and PWR_WUSR, can be read with privileged or unprivileged accesses.
- An unprivileged access to a privileged PWR bit or register is discarded: the bits are read as zero and the write to these bits is ignored (RAZ/WI).

11.9 PWR interrupts

Table 105 gives a summary of the interrupt sources and the way to control them.

Table 105. PWR interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep, Stop modes	Exit Standby retention mode	Exit Standby modes
WKUP	External WKUP	WUFx	WUPENx	CWUFx	Yes	Yes	Yes
WKUP_S(1)	External secure WKUP	WUFx	WUPENx	CWUFx	Yes	Yes	Yes
PVD	Programmable voltage detector through EXTI line 16	PVDO	EXTI line 16 enabled	EXTI PIF16	Yes	No	No

1. The WKUP_S secure interrupt is used only when TrustZone is enabled.

11.10 PWR registers

11.10.1 PWR control register 1 (PWR_CR1)

Address offset: 0x000

Reset value: 0x0000 0400 (reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB access

Access to this register can be protected by PWR LPMSEC and PWR SPRIV or PWR NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	R1RSB1	Res.	RTCRSB	RADIORSB	Res.	ULPMEN	R2RSB2	R2RSB1	Res.	Res.	LPMS[2:0]		
			rw		rw	rw		rw	rw	rw			rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **R1RSB1**: SRAM1 page 1, 64 KB retention in Standby mode

This bit is used to keep the SRAM1 page 1 content in Standby retention mode.

0: SRAM1 page 1 content not retained in Standby mode

1: SRAM1 page 1 content retained in Standby mode

Bit 11 Reserved, must be kept at reset value.

Bit 10 **RTCRSB**: RTC and TAMP retention in Standby mode

This bit is used to keep the RTC and TAMP functional in Standby retention mode.

0: RTC and TAMP content not retained in Standby mode (If any other SRAM retention in Standby mode bit (RxRSBy) is enabled, RTC and TAMP are also functional and retained in Standby mode)

1: RTC and TAMP functionality retained in Standby mode

Bit 9 **RADIORSB**: 2.4 GHz RADIO SRAMs (RXTXRAM and sequence RAM) and sleep clock retention in Standby mode

This bit is used to keep the 2.4 GHz RADIO SRAMs content in Standby retention mode and the 2.4 GHz RADIO sleep timer operational.

0: 2.4 GHz RADIO SRAMs and 2.4 GHz RADIO sleep timer content not retained in Standby mode

1: 2.4 GHz RADIO SRAMs and 2.4 GHz RADIO sleep timer content retained in Standby mode

Bit 8 Reserved, must be kept at reset value.

Bit 7 **ULPMEN**: BOR0 ultra-low-power mode

This bit is used to reduce the consumption by configuring the BOR0 in discontinuous mode for Stop 1, 2, 3 and Standby modes. Discontinuous mode is only available when BOR levels 1 to 4 and PVD are disabled.

0: BOR0 operating in continuous (normal) mode in all operating modes

1: BOR0 operating in discontinuous (ultra-low-power) mode in Stop 1, 2, 3, and Standby modes.

Note: This bit must be set to reach the lowest power consumption in the low-power modes.

Note: This bit must not be set together with autonomous peripherals using HSI16 as kernel clock.

Note: When BOR level 1 to 4 or PVD is enabled continuous mode applies independent from ULPMEN.

Bit 6 **R2RSB2**: SRAM2 page 2, 24 Kbyte retention in Standby mode

This bit is used to keep the SRAM2 page 2 content in Standby retention mode.

0: SRAM2 page 2 content not retained in Standby mode

1: SRAM2 page 2 content retained in Standby mode

Bit 5 **R2RSB1**: SRAM2 page 1, 8 Kbyte retention in Standby mode

This bit is used to keep the SRAM2 page 1 content in Standby retention mode.

0: SRAM2 page 1 content not retained in Standby mode

1: SRAM2 page 1 content retained in Standby mode

Bits 4:3 Reserved, must be kept at reset value.

Bits 2:0 **LPMS[2:0]**: Low-power mode selection

These bits select the low-power mode entered when the CPU enters the Deepsleep mode.

000: Stop 0 mode

001: Stop 1 mode

010: Stop 2 mode

011: Stop 3 mode

10x: Standby mode

Others: reserved

11.10.2 PWR control register 2 (PWR_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

Access to this register can be protected by PWR LPMSEC and PWR SPRIV or PWR NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FLASH FWU	Res.	PKARA MPDS	PRAM PDS	Res.	Res.	Res.	Res.	Res.	SRAM2 PDS2	SRAM2 PDS1	Res.	Res.	Res.	SRAM1 PDS1
	rw		rw	rw						rw	rw				rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **FLASHFWU**: Flash memory fast wake-up from Stop modes (Stop 0, 1)

This bit is used to obtain the best trade-off between low-power consumption and wake-up time when exiting the Stop 0 or Stop 1 modes.

When this bit is set, the flash memory remains in normal mode in Stop 0 and Stop 1 modes, which offers a faster startup time with higher consumption.

0: Flash memory enters low-power mode in Stop 0 and Stop 1 modes (lower-power consumption).

1: Flash memory remains in normal mode in Stop 0 and Stop 1 modes (faster wake-up time).

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PKARAMPDS**: PKA SRAM power-down in Stop 0 and Stop 1 modes

0: PKA SRAM content retained in Stop modes

1: PKA SRAM content lost in Stop modes

Note: The PKA SRAM content is lost in Stop 2, Stop 3, and Standby modes.

Bit 11 **PRAMPDS**: Peripheral USB power-down in Stop modes

0: Peripheral USNB_FS SRAM content retained in Stop modes

1: Peripheral USB SRAM content lost in Stop modes

Note: This bit is reserved on STM32WBA23xx devices

Note: The peripheral USB SRAM content is lost in Stop 2, Stop 3, and Standby modes.

Bits 10:6 Reserved, must be kept at reset value.

Bit 5 **SRAM2PDS2**: SRAM2 page 2, 24 Kbyte power-down in Stop modes

0: SRAM2 page 2 content retained in Stop modes

1: SRAM2 page 2 content lost in Stop modes

Note: The SRAM2 retention in Standby mode is controlled by R2RSB2 bit in PWR_CR1.

Bit 4 **SRAM2PDS1**: SRAM2 page 1, 8 Kbyte power-down in Stop modes

0: SRAM2 page 1 content retained in Stop modes

1: SRAM2 page 1 content lost in Stop modes

Note: The SRAM2 retention in Standby mode is controlled by R2RSB1 bit in PWR_CR1.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **SRAM1PDS1**: SRAM1 page 1, 64 Kbyte power-down in Stop modes

0: SRAM1 page 1 content retained in Stop modes

1: SRAM1 page 1 content lost in Stop modes

Note: The SRAM1 retention in Standby mode is controlled by R1RSB1 bit in PWR_CR1.

11.10.3 PWR control register 3 (PWR_CR3)

Address offset: 0x038

Power-on reset value: 0x0000 0000

Exit from Standby mode: not affected

System reset: not affected, except REGSEL

Access: 14 AHB clock cycles added compared to a standard AHB access

Access to this register can be protected by PWR VDMSEC and PWR SPRIV or PWR NSPRIV.

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

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **FSTEN**: Fast soft start

0: LDO/SMPS fast startup disabled (limited inrush current after system reset and wake-up from Standby modes)

1: LDO/SMPS fast startup enabled

Bit 1 **REGSEL**: Regulator selection

0: LDO selected

1: SMPS selected

Bit 0 Reserved, must be kept at reset value.

11.10.4 PWR voltage scaling register (PWR_VOSR)

Address offset: 0x00C

Reset value: 0x0000 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VOS[1:0]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VOSRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **VOS[1:0]**: Voltage scaling range selection

Set a and cleared by software.

Cleared by hardware when entering Stop 1, Stop 2, Stop 3, and Standby modes.

Access can be secured by RCC SYSCLKSEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.

00: Range 2 (lowest power)
01: Range 1 (highest frequency)
10: Range 1.5 (highest frequency, with lower power)
11: Reserved

Bit 15 **VOSRDY:** Ready bit for V_{CORE} voltage scaling output selection

Set and cleared by hardware. When decreasing the voltage scaling range, VOSRDY must be set before increasing the SYSCLK frequency.

0: Not ready, voltage level < VOS selected level
1: Ready, voltage level ≥ VOS selected level

Note: Following a write to VOS, it takes one hclk clock cycle for VOSRDY to be updated.

Bits 14:0 Reserved, must be kept at reset value.

11.10.5 PWR supply voltage monitoring control register (PWR_SVMCR)

Address offset: 0x010

Reset value: 0x0000 0000

Access to this register can be protected by PWR VDMSEC and PWR SPRIV or PWR NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	USV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDLS[2:0]			PVDE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **USV**: VDDUSB supply valid

This bit is used to validate the presence of a supply on VDDUSB for electrical and logical isolation purpose. Setting this bit is mandatory to use the USB peripheral.

0: VDDUSB not supplied, electrical and logical isolation enabled.
1: VDDUSB supply present, electrical and logical isolation disabled.

Note: This bit is reserved on STM32WBA23xx devices

Bits 27:8 Reserved, must be kept at reset value.

Bits 7:5 **PVDLS[2:0]**: Programmable voltage detector level selection
These bits select the voltage threshold detected by the programmable voltage detector:
000: V_{PVD0} ~ 2.0 V
001: V_{PVD1} ~ 2.2 V
010: V_{PVD2} ~ 2.4 V
011: V_{PVD3} ~ 2.5 V
100: V_{PVD4} ~ 2.6 V
101: V_{PVD5} ~ 2.8 V
110: V_{PVD6} ~ 2.9 V
111: Reserved.

Bit 4 **PVDE**: Programmable voltage detector enable
0: Programmable voltage detector disabled
1: Programmable voltage detector enabled

Bits 3:0 Reserved, must be kept at reset value.

11.10.6 PWR wake-up control register 1 (PWR_WUCR1)

Address offset: 0x014

Reset value: 0x0000 0000 (reset value not affected by exit from Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUPEN8	WUPEN7	WUPEN6	Res.	WUPEN4	WUPEN3	WUPEN2	WUPEN1
								rw	rw	rw		rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **WUPEN8**: Wake-up and interrupt pin WKUP8 enable
Access can be secured by PWR WUP8SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.
0: Wake-up and interrupt pin WKUP8 disabled
1: Wake-up and interrupt pin WKUP8 enabled

Bit 6 **WUPEN7**: Wake-up and interrupt pin WKUP7 enable
Access can be secured by PWR WUP7SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.
0: Wake-up and interrupt pin WKUP7 disabled
1: Wake-up and interrupt pin WKUP7 enabled

Bit 5 **WUPEN6**: Wake-up and interrupt pin WKUP6 enable
Access can be secured by PWR WUP6SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.
0: Wake-up and interrupt pin WKUP6 disabled
1: Wake-up and interrupt pin WKUP6 enabled

Bit 4 Reserved, must be kept at reset value.

Bit 3 **WUPEN4**: Wake-up and interrupt pin WKUP4 enable

Access can be secured by PWR WUP4SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.

0: Wake-up and interrupt pin WKUP4 disabled

1: Wake-up and interrupt pin WKUP4 enabled

Bit 2 **WUPEN3**: Wake-up and interrupt pin WKUP3 enable

Access can be secured by PWR WUP3SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.

0: Wake-up and interrupt pin WKUP3 disabled

1: Wake-up and interrupt pin WKUP3 enabled

Bit 1 **WUPEN2**: Wake-up and interrupt pin WKUP2 enable

Access can be secured by PWR WUP2SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.

0: Wake-up and interrupt pin WKUP2 disabled

1: Wake-up and interrupt pin WKUP2 enabled

Bit 0 **WUPEN1**: Wake-up and interrupt pin WKUP1 enable

Access can be secured by PWR WUP1SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.

0: Wake-up and interrupt pin WKUP1 disabled

1: Wake-up and interrupt pin WKUP1 enabled

11.10.7 PWR wake-up control register 2 (PWR_WUCR2)

Address offset: 0x018

Reset value: 0x0000 0000 (reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUPP8	WUPP7	WUPP6	Res.	WUPP4	WUPP3	WUPP2	WUPP1
								rw	rw	rw		rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **WUPP8**: Wake-up pin WKUP8 polarity

This bit must be configured when WUPEN8 = 0.

Access can be secured by PWR WUP8SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.

0: Detection on high level (rising edge)

1: Detection on low level (falling edge)

Bit 6 **WUPP7**: Wake-up pin WKUP7 polarity
This bit must be configured when WUPEN7 = 0.
Access can be secured by PWR WUP7SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 5 **WUPP6**: Wake-up pin WKUP6 polarity
This bit must be configured when WUPEN6 = 0.
Access can be secured by PWR WUP6SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 4 Reserved, must be kept at reset value.

Bit 3 **WUPP4**: Wake-up pin WKUP4 polarity
This bit must be configured when WUPEN4 = 0.
Access can be secured by PWR WUP4SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 2 **WUPP3**: Wake-up pin WKUP3 polarity
This bit must be configured when WUPEN3 = 0.
Access can be secured by PWR WUP3SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 1 **WUPP2**: Wake-up pin WKUP2 polarity
This bit must be configured when WUPEN2 = 0.
Access can be secured by PWR WUP2SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 0 **WUPP1**: Wake-up pin WKUP1 polarity.
This bit must be configured when WUPEN1 = 0.
Access can be secured by PWR WUP1SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

11.10.8 PWR wake-up control register 3 (PWR_WUCR3)

Address offset: 0x01C

Reset value: 0x0000 0000 (reset value not affected by exit from Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUSEL8[1:0]		WUSEL7[1:0]		WUSEL6[1:0]		Res.	Res.	WUSEL4[1:0]		WUSEL3[1:0]		WUSEL2[1:0]		WUSEL1[1:0]	
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **WUSEL8[1:0]**: Wake-up and interrupt pin WKUP8 selection

This field must be configured when WUPEN8 = 0.

Access can be secured by PWR WUP8SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.

00: reserved

01: WKUP8_1

10: WKUP8_2

11: WKUP8_3 (internal source, does not generate a WKUP interrupt)

Bits 13:12 **WUSEL7[1:0]**: Wake-up and interrupt pin WKUP7 selection

This field must be configured when WUPEN7 = 0.

Access can be secured by PWR WUP7SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.

00: reserved

01: WKUP7_1

10: reserved

11: WKUP7_3 (internal source, does not generate a WKUP interrupt)

Bits 11:10 **WUSEL6[1:0]**: Wake-up and interrupt pin WKUP6 selection

This field must be configured when WUPEN6 = 0.

Access can be secured by PWR WUP6SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.

00: WKUP6_0

01: WKUP6_1

10: reserved

11: WKUP6_3 (internal source, does not generate a WKUP interrupt)

Bits 9:8 Reserved, must be kept at reset value.

Bits 7:6 **WUSEL4[1:0]**: Wake-up and interrupt pin WKUP4 selection
This field must be configured when WUPEN4 = 0.
Access can be secured by PWR WUP4SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.
00: WKUP4_0
01: reserved
10: reserved
11: reserved

Bits 5:4 **WUSEL3[1:0]**: Wake-up and interrupt pin WKUP3 selection
This field must be configured when WUPEN3 = 0.
Access can be secured by PWR WUP3SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.
00: reserved
01: WKUP3_1
10: WKUP3_2
11: reserved

Bits 3:2 **WUSEL2[1:0]**: Wake-up and interrupt pin WKUP2 selection
This field must be configured when WUPEN2 = 0.
Access can be secured by PWR WUP2SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.
00: reserved
01: WKUP2_1
10: reserved
11: reserved

Bits 1:0 **WUSEL1[1:0]**: Wake-up and interrupt pin WKUP1 selection
This field must be configured when WUPEN1 = 0.
Access can be secured by PWR WUP1SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.
00: WKUP1_0
01: WKUP1_1
10: reserved
11: reserved

11.10.9 PWR disable backup domain register (PWR_DBPR)

Address offset: 0x028

Reset value: 0x0000 0000

Access to this register can be protected by PWR VBSEC and PWR SPRIV or PWR NSPRIV.

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

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **DBP**: Disable backup domain write protection

In reset state, all registers in the backup domain are protected against parasitic write access. This bit must be set to enable the write access to these registers. Before disabling backup domain access, make sure any write access to the domain has finished.

0: Write access to backup domain disabled

1: Write access to backup domain enabled

11.10.10 PWR security configuration register (PWR_SECCFGR)

Address offset: 0x030

Reset value: 0x0000 0000

When the system is secure (TZEN = 1), this register can be written only by a secure privileged access if SPRIV = 1 and by a secure privileged or unprivileged access if SPRIV = 0. A nonsecure write access generates an illegal access event and data is not written. This register can be read by secure or nonsecure, privileged, or unprivileged access.

When the system is not secure (TZEN = 0), this register is read as 0 and the register write is ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	VBSEC	VDMSEC	LPMSEC	Res.	Res.	Res.	Res.	WUP8SEC	WUP7SEC	WUP6SEC	Res.	WUP4SEC	WUP3SEC	WUP2SEC	WUP1SEC
	rw	rw	rw					rw	rw	rw		rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **VBSEC**: Backup domain secure protection

0: PWR_DBPR can be read and written with secure or nonsecure access.

1: PWR_DBPR can be read and written only with secure access.

Bit 13 **VDMSEC**: Voltage detection secure protection
0: PWR_SVMCR and PWR_CR3 can be read and written with secure or nonsecure access.
1: PWR_SVMCR and PWR_CR3 can be read and written only with secure access.

Bit 12 **LPMSEC**: Low-power modes secure protection
0: PWR_CR1, PWR_CR2 and CSSF in the PWR_SR can be read and written with secure or nonsecure access.
1: PWR_CR1, PWR_CR2, and CSSF in the PWR_SR can be read and written only with secure access.

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **WUP8SEC**: WUP8 secure protection
0: Bits related to the WKUP8 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written with secure or nonsecure access.
1: Bits related to the WKUP8 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written only with secure access.

Bit 6 **WUP7SEC**: WUP7 secure protection
0: Bits related to the WKUP7 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written with secure or nonsecure access.
1: Bits related to the WKUP7 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written only with secure access.

Bit 5 **WUP6SEC**: WUP6 secure protection
0: Bits related to the WKUP6 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written with secure or nonsecure access.
1: Bits related to the WKUP6 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written only with secure access.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **WUP4SEC**: WUP4 secure protection
0: Bits related to the WKUP4 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written with secure or nonsecure access.
1: Bits related to the WKUP4 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written only with secure access.

Bit 2 **WUP3SEC**: WUP3 secure protection
0: Bits related to the WKUP3 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written with secure or nonsecure access.
1: Bits related to the WKUP3 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written only with secure access.

Bit 1 **WUP2SEC**: WUP2 secure protection
0: Bits related to the WKUP2 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written with secure or nonsecure access.
1: Bits related to the WKUP2 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written only with secure access.

Bit 0 **WUP1SEC**: WUP1 secure protection
0: Bits related to the WKUP1 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written with secure or nonsecure access.
1: Bits related to the WKUP1 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written only with secure access.

11.10.11 PWR privilege control register (PWR_PRIVCFGR)

Address offset: 0x034

Reset value: 0x0000 0000

This register can be written only by a privileged access. It can be read by privileged or unprivileged access.

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **NSPRIV**: PWR nonsecure functions privilege configuration

This bit is set and reset by software.

It can be written only by privileged access, secure or nonsecure.

0: Read and write to PWR nonsecure functions can be done by privileged or unprivileged access.

1: Read and write to PWR nonsecure functions can be done by privileged access only.

Bit 0 **SPRIV**: PWR secure functions privilege configuration

This bit is set and reset by software.

It can be written only by a secure privileged access.

0: Read and write to PWR secure functions can be done by privileged or unprivileged access.

1: Read and write to PWR secure functions can be done by privileged access only.

11.10.12 PWR status register (PWR_SR)

Address offset: 0x038

Power-on reset value: 0x0000 0000

System reset value: 0b0000 0000 0000 0000 0000 0000 0000 0X00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP2F	SBF	STOPF	CSSF
												r	r	r	w

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **STOP2F**: Stop 2 and Stop 3 mode peripherals power down flag

This bit is set by hardware when the device enters Stop 2 or Stop 3 mode at the same time as the Stop 2 and Stop 3 mode peripherals are powered down by hardware. It's cleared by software by writing 1 to the CSSF bit.

0: The device did not enter Stop 2 or Stop 3 mode.

1: The device entered Stop 2 or Stop 3 mode.

Bit 2 **SBF**: Standby flag

This bit is set by hardware when the device enters the Standby mode and the CPU restart from its reset vector. It's cleared by writing 1 to the CSSF bit, or by a power-on reset. It is not cleared by the system reset.

0: The device did not enter Standby mode.

1: The device entered Standby mode.

Bit 1 **STOPF**: Stop flag

This bit is set by hardware when the device enters a Stop or Standby mode at the same time as the sysclk has been set by hardware to select HSI16. It's cleared by software by writing 1 to the CSSF bit and by hardware when SBF is set.

0: The device did not enter any Stop mode.

1: The device entered a Stop mode.

Bit 0 **CSSF**: Clear Stop and Standby flags

Access can be secured by PWR LPMSEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.

Writing 1 to this bit clears the STOPF, STOP2F, and SBF flags.

11.10.13 PWR supply voltage monitoring status register (PWR_SVMSR)

Address offset: 0x03C

Reset value: 0x0000 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACTVOS[1:0]	
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACTVOSRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDO	Res.	Res.	REGS	Res.
r											r			r	

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **ACTVOS**[1:0]: Voltage scaling range currently applied to V_{CORE}

This field provides the last VOS value.

00: Range 2 (lowest power)

01: Range 1 (highest frequency)

10: Range 1.5 (highest frequency, lower power)

11: Reserved

Bit 15 **ACTVOSRDY**: Voltage level ready for currently used VOS
0: V_{CORE} is above or below the current voltage scaling provided by ACTVOS.
1: V_{CORE} is equal to the current voltage scaling provided by ACTVOS
Note: Following a write to VOS, it takes one hclk clock cycle for ACTVOSRDY to be updated.

Bits 14:5 Reserved, must be kept at reset value.

Bit 4 **PVDO**: Programmable voltage detector output
0: V_{DD} is equal or above the PVD threshold selected by PVDLS[2:0].
1: V_{DD} is below the PVD threshold selected by PVDLS[2:0].

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **REGS**: Regulator selection
0: LDO selected
1: SMPS selected

Bit 0 Reserved, must be kept at reset value.

11.10.14 PWR wake-up status register (PWR_WUSR)

Address offset: 0x044

Reset value: 0x0000 0000

(Reset value not affected by exit from Standby modes)

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUF8	WUF7	WUF6	Res.	WUF4	WUF3	WUF2	WUF1
								r	r	r		r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **WUF8**: Wake-up and interrupt pending flag 8
This bit is set when a wake-up event is detected on WKUP8 pin. This bit is cleared by writing 1 in the CWUF8 bit of PWR_WUSCR when WUSEL8 ≠ 11, or by hardware when WUPEN8 = 0.
When WUSEL8 = 11, this bit is cleared by hardware when all associated internal wake-up source are cleared.
When WUSEL8 = 11, no WKUP interrupt is generated

Bit 6 **WUF7**: Wake-up and interrupt pending flag 7
This bit is set when a wake-up event is detected on WKUP7 pin. This bit is cleared by writing 1 in the CWUF7 bit of PWR_WUSCR when WUSEL7 ≠ 11, or by hardware when WUPEN7 = 0.
When WUSEL7 = 11, this bit is cleared by hardware when all associated internal wake-up source are cleared.
When WUSEL7 = 11, no WKUP interrupt is generated.

Bit 5 **WUF6**: Wake-up and interrupt pending flag 6

This bit is set when a wake-up event is detected on WKUP6 pin. This bit is cleared by writing 1 in the CWUF6 bit of PWR_WUSCR when WUSEL6 ≠ 11, or by hardware when WUPEN6 = 0.
When WUSEL6 = 11, this bit is cleared by hardware when all associated internal wake-up source are cleared.
When WUSEL6 = 11, no WKUP interrupt is generated

Bit 4 Reserved, must be kept at reset value.

Bit 3 **WUF4**: Wake-up and interrupt pending flag 4

This bit is set when a wake-up event is detected on WKUP4 pin. This bit is cleared by writing 1 in the CWUF4 bit of PWR_WUSCR or by hardware when WUPEN4 = 0.

Bit 2 **WUF3**: Wake-up and interrupt pending flag 3

This bit is set when a wake-up event is detected on WKUP3 pin. This bit is cleared by writing 1 in the CWUF3 bit of PWR_WUSCR or by hardware when WUPEN3 = 0.

Bit 1 **WUF2**: Wake-up and interrupt pending flag 2

This bit is set when a wake-up event is detected on WKUP2 pin. This bit is cleared by writing 1 in the CWUF2 bit of PWR_WUSCR or by hardware when WUPEN2 = 0.

Bit 0 **WUF1**: Wake-up and interrupt pending flag 1

This bit is set when a wake-up event is detected on WKUP1 pin. This bit is cleared by writing 1 in the CWUF1 bit of PWR_WUSCR or by hardware when WUPEN1 = 0.

11.10.15 PWR wake-up status clear register (PWR_WUSCR)

Address offset: 0x048

Reset value: 0x0000 0000

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CWUF8	CWUF7	CWUF6	Res.	CWUF4	CWUF3	CWUF2	CWUF1
								w	w	w		w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **CWUF8**: Clear wake-up flag 8

Access can be secured by PWR WUP8SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.
Writing 1 to this bit clears the WUF8 flag in PWR_WUSR.

Bit 6 **CWUF7**: Clear wake-up flag 7

Access can be secured by PWR WUP7SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.
Writing 1 to this bit clears the WUF7 flag in PWR_WUSR.

Bit 5 **CWUF6**: Clear wake-up flag 6

Access can be secured by PWR WUP6SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV. Writing 1 to this bit clears the WUF6 flag in PWR_WUSR.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **CWUF4**: Clear wake-up flag 4

Access can be secured by PWR WUP4SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV. Writing 1 to this bit clears the WUF4 flag in PWR_WUSR.

Bit 2 **CWUF3**: Clear wake-up flag 3

Access can be secured by PWR WUP3SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV. Writing 1 to this bit clears the WUF3 flag in PWR_WUSR.

Bit 1 **CWUF2**: Clear wake-up flag 2

Access can be secured by PWR WUP2SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV. Writing 1 to this bit clears the WUF2 flag in PWR_WUSR.

Bit 0 **CWUF1**: Clear wake-up flag 1

Access can be secured by PWR WUP1SEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV. Writing 1 to this bit clears the WUF1 flag in PWR_WUSR.

11.10.16 PWR port A Standby I/O retention enable register (PWR_IORETENRA)

Address offset: 0x050

Reset value: 0x0000 0000 (reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	EN7	EN6	EN5	Res.	Res.	EN2	EN1	EN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:5 **EN[15:5]**: Port A Standby GPIO pin y (y = 15 to 5) retention enable

Access can be secured by GPIOA SECy. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.

0: PAy Standby GPIO retention feature disabled.

1: PAy Standby GPIO retention feature enabled.

Bits 4:3 Reserved, must be kept at reset value.

Bits 2:0 **EN[2:0]**: Port A Standby GPIO pin y (y = 2 to 0) retention enable

Note: Bit 0 is reserved on STM32WBA25xx devices.

11.10.17 PWR port A Standby I/O retention status register (PWR_IORETRA)

Address offset: 0x054

Reset value: 0x0000 0000 (reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RET15	RET14	RET13	RET12	RET11	RET10	RET9	RET8	RET7	RET6	RET5	Res.	Res.	RET2	RET1	RET0
rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0			rc_w0	rc_w0	rc_w0

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:5 **RET[15:5]**: Port A Standby GPIO pin y (y = 15 to 5) retention enable

Access can be secured by GPIOA SECy. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.

1: Set by hardware when Standby GPIO PAy is enabled in PWR_IORETENRA and Standby mode is entered. Standby GPIO retention PAy active.

0: Cleared by software, writing 0. Standby GPIO retention PAy disabled.

Bits 4:3 Reserved, must be kept at reset value.

Bits 2:0 **RET[2:0]**: Port A Standby GPIO pin y (y = 2 to 0) retention enable

Note: Bit 0 is reserved on STM32WBA25xx devices.

11.10.18 PWR port B Standby I/O retention enable register

(PWR_IORETENRB)

Address offset: 0x058

Reset value: 0x0000 0000 (reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	Res.	Res.	EN12	Res.	Res.	EN9	EN8	Res.	EN6	EN5	EN4	EN3	EN2	Res.	Res.
rw			rw			rw	rw		rw	rw	rw	rw	rw		

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EN[15]**: Port B Standby GPIO pin y (y = 15) retention enable
Access can be secured by GPIOB SECy. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.
0: PBy Standby GPIO retention feature disabled.
1: PBy Standby GPIO retention feature enabled.

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **EN[12]**: Port B Standby GPIO pin y (y = 12) retention enable

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **EN[9:8]**: Port B Standby GPIO pin y (y = 9 to 8) retention enable

Bit 7 Reserved, must be kept at reset value.

Bits 6:2 **EN[6:2]**: Port B Standby GPIO pin y (y = 6 to 2) retention enable

Bits 1:0 Reserved, must be kept at reset value.

11.10.19 PWR port B Standby I/O retention status register (PWR_IORETRB)

Address offset: 0x05C

Reset value: 0x0000 0000 (reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RET15	Res.	Res.	RET12	Res.	Res.	RET9	RET8	Res.	RET6	RET5	RET4	RET3	RET2	Res.	Res.
rc_w0			rc_w0			rc_w0	rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0		

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **RET[15]**: Port B Standby GPIO pin y (y = 15) retention enable
Access can be secured by GPIOB SECy. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.
1: Set by hardware when Standby GPIO PBy is enabled in PWR_IORETENRB and Standby mode is entered. Standby GPIO retention PBy active.
0: Cleared by software, writing 0. Standby GPIO retention PBy disabled.

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **RET[12]**: Port B Standby GPIO pin y (y = 12) retention enable

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **RET[9:8]**: Port B Standby GPIO pin y (y = 9 to 8) retention enable

Bit 7 Reserved, must be kept at reset value.

Bits 6:2 **RET[6:2]**: Port B Standby GPIO pin y (y = 6 to 2) retention enable

Bits 1:0 Reserved, must be kept at reset value.

11.10.20 PWR port C Standby I/O retention enable register (PWR_IORETENRC)

Address offset: 0x060

Reset value: 0x0000 0000

(Reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	EN14	EN13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **EN[15:13]**: Port C Standby GPIO pin y (y = 15 to 13) retention enable
Access can be secured by GPIOC SECy. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.
0: PCy Standby GPIO retention feature disabled.
1: PCy Standby GPIO retention feature enabled.

Bits 12:0 Reserved, must be kept at reset value.

11.10.21 PWR port C Standby I/O retention status register

(PWR_IORETRC)

Address offset: 0x064

Reset value: 0x0000 0000

(Reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RET15	RET14	RET13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rc_w0	rc_w0	rc_w0													

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **RET[15:13]**: Port C Standby GPIO pin y (y = 15 to 13) retention enable

Access can be secured by GPIOC SECy. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.

1: Set by hardware when Standby GPIO PCy is enabled in PWR_IORETENRC and Standby mode is entered. Standby GPIO retention PCy active.

0: Cleared by software, writing 0. Standby GPIO retention PCy disabled.

Bits 12:0 Reserved, must be kept at reset value.

11.10.22 PWR port H Standby I/O retention enable register (PWR_IORETENRH)

Address offset: 0x088

Reset value: 0x0000 0000

(Reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN3	Res.	Res.	Res.
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **EN3**: Port H Standby GPIO pin y (y = 3) retention enable

Access can be secured by GPIOH SECy. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.

0: PHy Standby GPIO retention feature disabled.

1: PHy Standby GPIO retention feature enabled.

Bits 2:0 Reserved, must be kept at reset value.

11.10.23 PWR port H Standby I/O retention status register (PWR_IORETRH)

Address offset: 0x08C

Reset value: 0x0000 0000

Reset value is not affected by exit Standby mode.

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RET3	Res.	Res.	Res.
												rc_w0			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RET3**: Port H Standby GPIO pin y (y = 3) retention enable

Access can be secured by GPIOH SECy. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.

1: Set by hardware when Standby GPIO PHy is enabled in PWR_IORETENRH and Standby mode is entered. Standby GPIO retention PHy active.

0: Cleared by software, writing 0. Standby GPIO retention PHy disabled.

Bits 2:0 Reserved, must be kept at reset value.

11.10.24 PWR 2.4 GHz RADIO status and control register (PWR_RADIOSCR)

Address offset: 0x100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REGPA BYPEN	REGPA SEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw	rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REGPA RDYVD DRFPA	Res.	Res.	RFVDDHPA[4:0]					Res.	Res.	Res.	Res.	ENCM ODE	PHYM ODE	MODE[1:0]	
r			r	r	r	r	r					r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **REGPABYPEN**: regulator REG_VDDHPA bypass enable

This bit must be written only when the VDDHPA regulator is not used.

When REGPASEL = 1 this bit has no meaning.

When REGPASEL = 0, bypasses the REG_VDDHPA regulator when VDDHPA 1.2 V is requested and input voltage is VDD11.

Access can be secured by GTZC1_TZSC RADIOSEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.

0: 2.4 GHz RADIO PA supplied by REG_VDDHPA regulator output voltage.

1: 2.4 GHz RADIO PA 1.2 V supplied directly from internal V_{DD11} (available only when REGPASEL = 0)

Bit 23 **REGPASEL**: regulator REG_VDDHPA input supply selection

This bit must be written only when the VDDHPA regulator is not used.

Access can be secured by GTZC1_TZSC RADIOSEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.

0: VDDRFPA pin selected as regulator REG_VDDHPA input supply.

1: regulator REG_VDDHPA input supply selection between V_{DDRFPA} and V_{DD11} depending on the requested regulated output voltage. see *Table 84: 2.4 GHz RADIO supply configuration*. Must be set only when device SMPS is used to generate V_{DD11}.

Bits 22:16 Reserved, must be kept at reset value.

Bit 15 **REGPARDYVDDRFPA:** Ready bit for V_{DDHPA} voltage level when selecting VDDRFPA input

0: Not ready, V_{DDHPA} voltage level < REGPAVOS selected supply level

1: Ready, V_{DDHPA} voltage level ≥ REGPAVOS selected supply level

Note: REGPARDYVDDRFPA does not allow to detect correct V_{DDHPA} voltage level when request to lower the level.

Bits 14:13 Reserved, must be kept at reset value.

Bits 12:8 **RFVDDHPA[4:0]**: 2.4 GHz RADIO VDDHPA control word

Bits [3:0] see *Table 83: PA output power table format* for definition.

Bit [4] rf_event.

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **ENCMODE**: 2.4 GHz RADIO encryption function operating mode

0: 2.4 GHz RADIO encryption function disabled

1: 2.4 GHz RADIO encryption function enabled

Bit 2 **PHYMODE**: 2.4 GHz RADIO PHY operating mode

0: 2.4 GHz RADIO Sleep mode

1: 2.4 GHz RADIO Standby mode

Bits 1:0 **MODE[1:0]**: 2.4 GHz RADIO operating mode.

00: 2.4 GHz RADIO Deepsleep mode

01: 2.4 GHz RADIO Sleep mode

1x: 2.4 GHz RADIO Active mode

11.10.25 PWR Stop 2 peripheral IOs retention register (PWR_S2RETR)

Address offset: 0x104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTASR
															rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTASREN
															rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **PTASR**: PTA interface output signals state retention in Stop 2 and 3 mode active

Access can be secured by GTZC1_TZSC PTACONVSEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.

1: Set by hardware when Stop 2 and Stop 3 mode PTA retention is enabled in PTASREN and Stop 2 and Stop 3 mode is entered. Stop 2 and Stop 3 PTA output signals retention active.

0: Cleared by software, writing 0. Stop 2 and Stop 3 PTA output signals retention disabled.

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **PTASREN**: PTA output signals Stop 2 and Stop 3 mode retention enable

Access can be secured by GTZC1_TZSC PTACONVSEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with PWR SPRIV or when nonsecure with PWR NSPRIV.

0: PTA output signals Stop 2 and Stop 3 retention feature disabled.

1: PTA output signals Stop 2 and Stop 3 retention feature enabled.

11.10.26 PWR register map

Table 106. PWR register map and reset values

Offset	Register	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
0x000	PWR_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	R1RSB1	Res.	RTCRSB	RADIORSB	Res.	ULPMEN	R2RSB2	R2RSB1	Res.	Res.	LPMS [2:0]		
	Reset value																				0		1	0		0	0	0			0	0	0
0x004	PWR_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLASHFWU	Res.	PKARAMPDS	PRAMPDS	Res.	Res.	Res.	Res.	Res.	SRAM2PDS2	SRAM2PDS1	Res.	Res.	Res.	SRAM1PDS1
	Reset value																		0		0	0						0	0				0
0x008	PWR_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSTEN	REGSEL	Res.
	Reset value																														0	0	
0x00C	PWR_VOSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VOS[1:0]		VOSRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0	1															
0x010	PWR_SVMCR	Res.	Res.	Res.	USV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDLS [2:0]			PVDE	Res.	Res.	Res.	Res.
	Reset value				0																					0	0	0	0				
0x014	PWR_WUCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUPEN8	WUPEN7	WUPEN6	Res.	WUPEN4	WUPEN3	WUPEN2	WUPEN1
	Reset value																									0	0	0		0	0	0	0
0x018	PWR_WUCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUPP8	WUPP7	WUPP6	Res.	WUPP4	WUPP3	WUPP2	WUPP1
	Reset value																									0	0	0		0	0	0	0
0x01C	PWR_WUCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUSEL8[1:0]		WUSEL7[1:0]		WUSEL6[1:0]		Res.	Res.	WUSEL4[1:0]		WUSEL3[1:0]		WUSEL2[1:0]		WUSEL1[1:0]	
	Reset value																	0	0	0	0	0	0			0	0	0	0	0	0	0	0
0x020 to 0x024	Reserved																																
0x028	PWR_DBPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBP
	Reset value																																0
0x02C	Reserved																																
0x030	PWR_SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBSEC	VDMSEC	LPMSEC	Res.	Res.	Res.	Res.	WUP8SEC	WUP7SEC	WUP6SEC	Res.	WUP4SEC	WUP3SEC	WUP2SEC	WUP1SEC
	Reset value																		0	0	0					0	0	0		0	0	0	0
0x034	PWR_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NSPRIV	SPRIV
	Reset value																															0	0

Table 106. PWR register map and reset values (continued)

Offset	Register	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
0x038	PWR_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP2F	SBF	STOPF	CSSF
	Reset value																													0	0	0	0
0x03C	PWR_SVMSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACTVOS[1:0]		ACTVOSRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDO	Res.	Res.	REGS	Res.
	Reset value															0	0	1											0			0	
0x040	Reserved																																
0x044	PWR_WUSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUF8	WUF7	WUF6	Res.	WUF4	WUF3	WUF2	WUF1
	Reset value																									0	0	0		0	0	0	0
0x048	PWR_WUSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CWUF8	CWUF7	CWUF6	Res.	CWUF4	CWUF3	CWUF2	CWUF1
	Reset value																									0	0	0		0	0	0	0
0x04C	Reserved																																
0x050	PWR_IORETENRA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	EN7	EN6	EN5	Res.	Res.	EN2	EN1	EN0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0			0	0	0
0x054	PWR_IORETRA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RET15	RET14	RET13	RET12	RET11	RET10	RET9	RET8	RET7	RET6	RET5	Res.	Res.	RET2	RET1	RET0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0			0	0	0
0x058	PWR_IORETENRB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN15	Res.	Res.	EN12	Res.	Res.	EN9	EN8	Res.	EN6	EN5	EN4	EN3	EN2	Res.	Res.
	Reset value																	0			0			0	0		0	0	0	0	0		
0x05C	PWR_IORETRB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RET15	Res.	Res.	RET12	Res.	Res.	RET9	RET8	Res.	RET6	RET5	RET4	RET3	RET2	Res.	Res.
	Reset value																	0			0			0	0		0	0	0	0	0		
0x060	PWR_IORETENRC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN15	EN14	EN13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0	0													
0x064	PWR_IORETRC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RET15	RET14	RET13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0	0													
0x068 to 0x084	Reserved																																
0x088	PWR_IORETENRH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN3	Res.	Res.	Res.
	Reset value																													0			
0x08C	PWR_IORETRH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RET3	Res.	Res.	Res.
	Reset value																													0			
0x090 to 0x0FC	Reserved																																

Table 106. PWR register map and reset values (continued)

Offset	Register	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
0x100	PWR_RADIOSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REGPABYPEN	REGPASEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REGPARDYVDDRFPA	Res.	Res.	RFVDDHPA[4:0]					Res.	Res.	Res.	Res.	ENCMODE	PHYMODE	MODE[1:0]	
	Reset value								0	0								0			0	0	0	0	0					0	0	0	0
0x104	PWR_S2RETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTASR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTASREN
	Reset value																0																0
0x108 to 0x3FC	Reserved																																

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

12 Reset and clock control (RCC)

12.1 Introduction

The reset and clock control (RCC) manages the different kinds of reset and generates all clocks for the bus and peripherals.

12.2 RCC pins and internal signals

Table 107 lists the RCC inputs and output signals connected to package pins or balls.

Table 107. RCC input/output signals connected to package pins or balls

Signal name	Signal type	Description
NRST	I/O	System reset, can be used to provide reset to external devices
OSC32_IN	I	32 kHz oscillator input
OSC32_OUT	O	32 kHz oscillator output
OSC_IN	I	System oscillator input
OSC_OUT	O	System oscillator output
MCO	O	Output clock for external devices
LSCO	O	Low-speed output clock for external devices
AUDIOCLK	I	External kernel clock input for SAI1

12.3 RCC reset functional description

The following types of reset exist:

- Power reset
- System reset
- Backup domain reset
- Individual peripheral reset

12.3.1 Power reset

A power reset is generated when one of the following events occurs:

- Upon brownout reset (BOR)
- When exiting Standby modes

A BOR sets all registers to their reset values. Five BOR threshold levels can be selected through the user option bytes. The backup domain is always reset on the V_{BOR0} threshold, as is the power-on reset.

When exiting Standby mode, all registers in the core domain are set to their reset value. Registers outside the core domain (RTC, TAMP, WKUP, IWDG, and Standby modes control) are not impacted.

12.3.2 System reset

A system reset sets all registers to their reset values, except the reset flags in the *RCC control/status register (RCC_CSR)* and the registers in the backup domain.

A system reset is generated when one of the following events occurs:

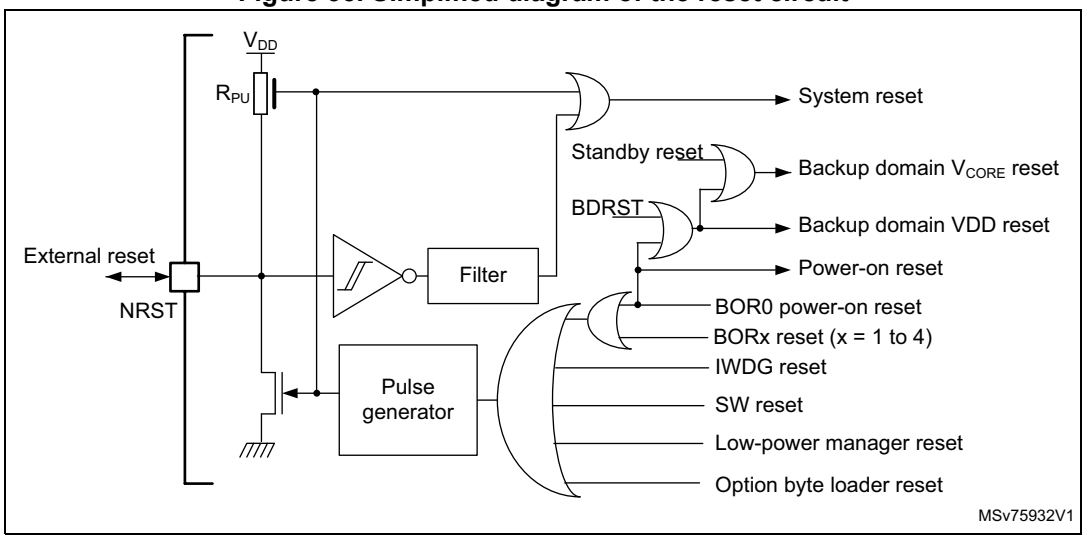
- A low level on the NRST pin (external reset)
- An independent watchdog event (IWDG reset)
- A software reset (SW reset) (see *Software reset*)
- A low-power mode security reset (see *Low-power mode security reset*)
- An option byte loader reset (see *Option byte loader reset*)
- A brownout reset

The reset source can be identified by checking the reset flags in the *RCC control/status register (RCC_CSR)*.

The device internal reset sources (such as BOR and IWDG) act on the NRST pin, which is always kept low during the delay phase. The internal reset signal is output on the NRST pin. The pulse generator ensures a minimum reset pulse duration of $t_{O(NRST)}$ for each internal reset source (refer to the product datasheet for more details). In case of an external reset, the reset pulse is generated while the NRST pin is asserted low.

In case of an internal reset, the internal pull-up R_{PU} is deactivated to save the power consumption through the pull-up resistor.

Figure 35. Simplified diagram of the reset circuit



Software reset

The SYSRESETREQ bit in the core application interrupt and reset control register must be set to force a software reset on the device.

Low-power mode security reset

To avoid that critical applications mistakenly enter a low-power mode, low-power mode security resets are available. If enabled in option bytes, a reset is generated in any of the following conditions:

- Entering Stop mode: this type of reset is enabled by resetting the nRST_STOP bit in the user option bytes. In this case, whenever a Stop mode entry sequence is successfully executed, the device is reset instead of entering Stop mode.
- Entering Standby mode: this type of reset is enabled by resetting the nRST_STDBY bit in the user option bytes. In this case, whenever a Standby mode entry sequence is successfully executed, the device is reset instead of entering Standby mode.

For further information on the user option bytes, refer to *Section 7.4.1: Option bytes description*.

Option byte loader reset

The option byte loader reset is generated when the OBL_LAUNCH bit is set in the FLASH_CR register. This bit is used to launch the option byte loading by software.

12.3.3 Backup domain reset

The backup domain has two specific resets.

A backup domain reset is generated when one of the following events occurs:

- A software reset, triggered by setting the BDRST bit in the *RCC backup domain control register (RCC_BDCR1)*
- A V_{DD} supply BOR0 power-on reset
- A V_{CORE} supply power-on reset (only resetting the backup domain V_{CORE}: RTC, TAMP, and backup registers).

A backup domain reset only affects the LSE oscillator, the RTC, TAMP, and backup registers, and the RCC_BDCRx registers.

12.3.4 Individual peripheral reset

Individual peripherals can be reset by software with their reset register bit in the RCC.

12.3.5 CPU reset

The CPU reset vector is selected through the boot option bytes.

12.4 RCC clocks functional description

The following clock sources can be used to drive the system clock (SYSCLK):

- HSI16: high-speed internal 16 MHz RC oscillator clock
- HSE32: high-speed external crystal or clock, 32 MHz
- PLL1: clock up to 64 MHz

The HSI16 is used as the system clock source after startup from reset and is configured at 16 MHz.

The device has the following additional clock sources:

- LSI:
  - LSI1: 32 kHz low-speed, low-power internal RC that drives the independent watchdog and, optionally, the RTC used for auto-wake-up from Stop 0, 1, 2, and Standby modes
  - LSI2: 32 kHz low-speed low drift internal RC that drives optionally the RTC or 2.4 GHz RADIO sleep timer used for auto-wake-up from Stop and Standby modes (not available on STM32WBA25/23 devices).
- LSE: 32.768 kHz low-speed external crystal or clock that optionally drives the real-time clock (rtc_ck)

Each clock source can be switched on or off independently when it is not used, to optimize power consumption.

The frequencies for AHB1, AHB2, AHB4, and AHB5 for the PTACONV, as well as APB1, APB2, and APB7 are derived from the SYSCLK and are provided with several prescalers to configure them. The maximum frequency of these AHB and APB domains is 64 MHz.

The AHB5 frequency for the 2.4 GHz RADIO is also derived from the SYSCLK and is provided with a prescaler to configure it. The maximum frequency of this AHB5 domain is 32 MHz.

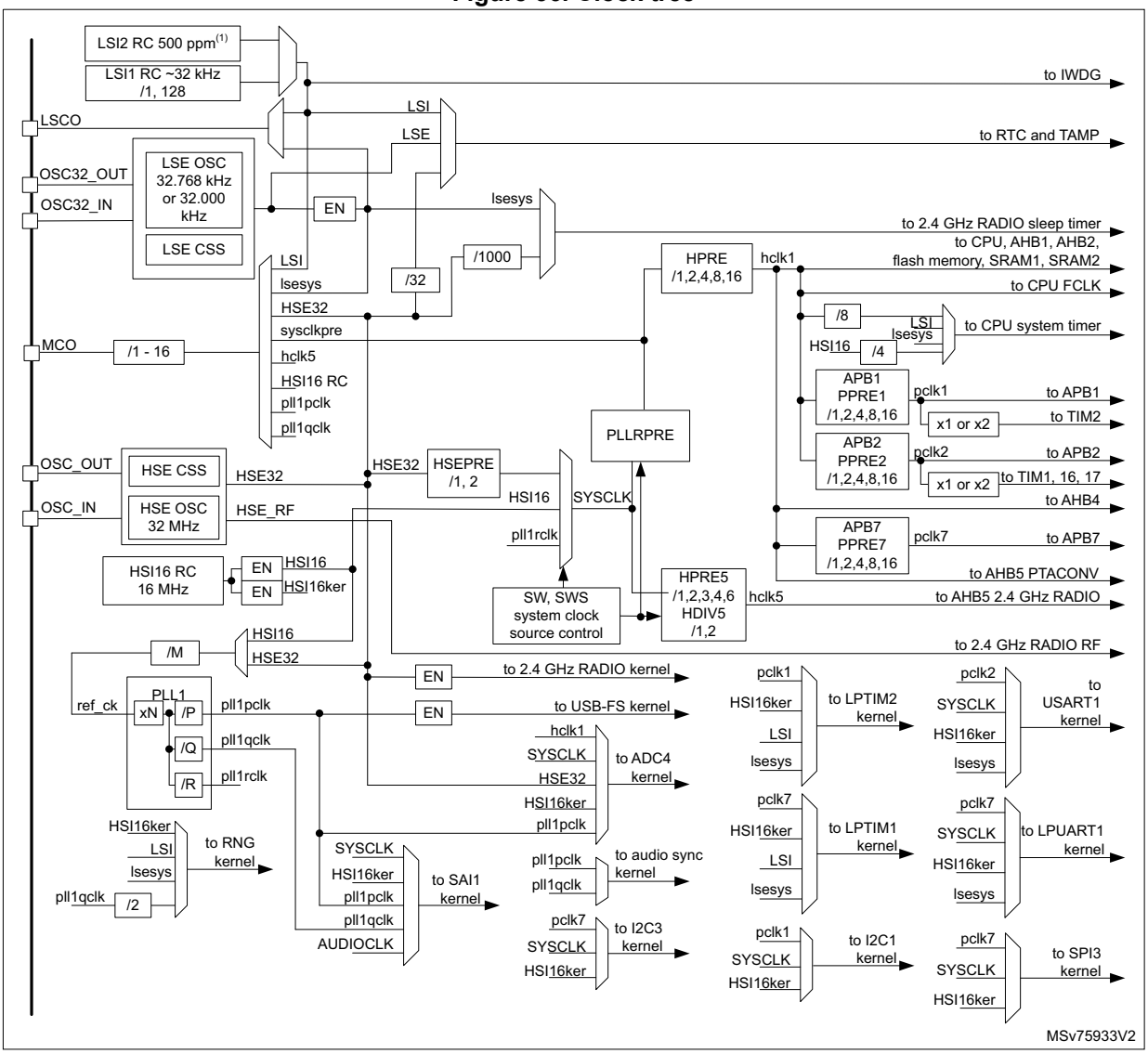
Most peripheral kernel clocks are derived from their bus clock (hclk1, hclk5, pclk1, pclk2, or pclk7). In addition, some peripherals receive an independent kernel clock. For these peripherals the kernel clock can be selected by software between several sources through the RCC_CCIPRx registers (x = 1, 2, 3):

- RNG
- ADC4
- U(S)ART1
- LPUART1
- I2Cx (x = 1, 3)
- SPI3
- SAI1
- USB
- LPTIMx (x = 1, 2)
- RTC and TAMP (selected in RCC_BDCR1)
- IWDG (always LSI)
- 2.4 GHz RADIO (always HSE32)
- 2.4 GHz RADIO sleep timer (selected in RCC_BDCR1)
- RCC audio synchronization

The RCC feeds the CPU system timer (SysTick) clock with the AHB clock (hclk1) divided by 8, HSI16 divided by 4, LSE, or LSI. The SysTick can work either with this clock or directly with the CPU bus clock (hclk1), configurable in the CPU SysTick control and status register.

FCLK acts as the CPU free-running clock.

Figure 36. Clock tree



12.4.1 HSE32 clock with trimming

The HSE32 32 MHz external oscillator has the advantage of producing a very accurate rate on the main clock. The HSE32 furthermore provides an on-chip trimming capability in RCC_ECSCR1. The HSE32 must be used for any 2.4 GHz RADIO transmission and reception.

The high-speed external clock signal (HSE32) can be generated from the following clock sources:

- HSE32 external crystal
- HSE32 external clock

The clock source must be placed as close as possible to the oscillator pins to minimize output distortion and startup stabilization time.

HSE32 is controlled from the CPU and from the 2.4 GHz RADIO.

HSE32 can be switched on and off using the HSEON bit in the *RCC clock control register (RCC_CR)*. HSE32 must be enabled with the HSEON bit when used for the CPU.

The HSERDY flag in the *RCC clock control register (RCC_CR)* indicates if the HSE32 oscillator is stable and forwarded or not for use by the CPU. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *RCC clock interrupt enable register (RCC_CIER)*. Refer to the product datasheet for the HSE32 stabilization-ready time.

The 2.4 GHz RADIO when waking up from a sleep timer event enables HSE32 autonomously for its own purpose with STRADIOCLKON, independently of the HSEON bit. To use the 2.4 GHz RADIO outside a sleep timer event, software has to enable HSE32 with HSEON.

External crystal (HSE32 crystal)

The associated hardware configuration is shown in *Figure 37*. Refer to the electrical characteristics section of the datasheet for more details.

Frequency trimming

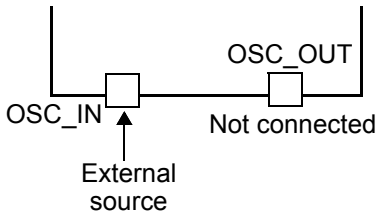
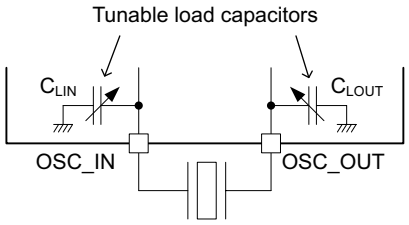
When using HSE32 with the external crystal, the load capacitors are provided by the integrated capacitor banks, which can be trimmed. The HSE32 load capacitor trimming allows a compensation of device manufacturing process variations, used crystal, and PCB design. The HSE32 frequency can be tuned in the application through the RCC_ECSCR1 register.

The HSE32 frequency can be measured by outputting the HSE32 clock on the MCO when in Run or Sleep mode.

External clock (HSE32 external)

The associated hardware configuration is shown in *Figure 37*. Refer to the electrical characteristics section of the datasheet for more details.

Figure 37. HSE32 hardware configurations

Clock source	Hardware configuration
External clock	
Crystal	

12.4.2 HSI16 clock

The HSI16 clock signal is generated from an internal 16 MHz RC oscillator.

The HSI16 RC oscillator has the advantage of providing a clock source at low cost. It also has a faster startup time than the HSE32 crystal oscillator. However, even with calibration, the frequency is less accurate than an external crystal oscillator.

The HSI16 clock is used as the system clock after reset and wake-up from Stop and Standby modes. It can also be used as a backup clock source (auxiliary clock) for the CPU if the HSE32 crystal oscillator fails. Refer to *Section 12.4.8: HSE32 clock security system (HSECSS)*.

The HSIRDY flag in the *RCC clock control register (RCC_CR)* indicates if the HSI16 RC oscillator is stable or not. At startup, the HSI16 clock is not released until this bit is set by hardware.

The HSI16 RC can be switched on and off using the HSION bit in the *RCC clock control register (RCC_CR)*.

Calibration

The RC oscillator frequencies may vary from one chip to another due to manufacturing process variations, which is why each device is factory-calibrated by STMicroelectronics for 1 % accuracy at T_A = 30°C.

After reset, the factory calibration value is loaded in the HSICAL bits in the *RCC internal clock sources calibration register 3 (RCC_ICSCR3)*.

If the application is subject to voltage or temperature variations, this may affect the RC oscillator speed. The HSI16 frequency can be trimmed in the application using the HSITRIM in the *RCC internal clock sources calibration register 3 (RCC_ICSCR3)*.

For more details on how to measure the HSI16 frequency variation, refer to *Section 12.4.20*.

12.4.3 PLL1

The RCC features one PLL1 that is generally used to provide clocks to the CPU and to some peripherals. The PLL1 must be enabled only in range 1 and range 1.5. The PLL must be disabled before entering range 2.

The PLL1 integrated in the RCC offers the following features:

- Input frequency range: 4 to 16 MHz
- VCO frequency range: 128 to 544 MHz
- Integer and fractional modes
- A 13-bit sigma-delta (ΣΔ) modulator for fine-tuning the VCO frequency by steps of 11 to 0.3 ppm

  The ΣΔ modulator can be updated on the fly, without generating frequency overshoots on PLL1 outputs.
- Three outputs with postdividers

The PLL1 is controlled via the RCC_PLL1DIVR, RCC_PLL1FRACR, RCC_PLL1CFGR, and RCC_CR registers.

The frequency of the reference clock provided to the PLL1 (F_{ref_ck}) must range between 4 and 16 MHz. The user application must program the PLL1M dividers properly in the *RCC PLL1 configuration register (RCC_PLL1CFGR)* to match this condition. Additionally, PLL1RGE must be set according to the reference input frequency to ensure an optimal performance of the PLL1.

To reduce power consumption, it is recommended to configure the VCO output to the lowest possible frequency.

The PLL1N loop divider must be programmed to achieve the expected frequency at VCO output. In addition, the VCO output range (128 to 544 MHz) must be respected.

The PLL1 operates in integer mode when PLL1FRACEN is 0 and the PLL is enabled with PLL1ON. Fractional mode can be enabled at any time by setting PLL1FRACN to the required value, and subsequently setting the PLL1FRACEN bit. The ΣΔ modulator is designed to minimize the jitter impact while allowing very-small-step frequency adjustments. To update the fractional value, first clear the PLL1FRACEN bit before updating the PLL1FRACN value, and subsequently set PLL1FRACEN again. The old PLL1FRACN value is used until the new value is activated by setting PLL1FRACEN. PLL1FRACN can be updated by software only when PLL1FRACEN has been cleared.

The PLL1 can be enabled by setting PLL1ON to 1 in the *RCC clock control register (RCC_CR)*. The PLL1RDY bit indicates that the PLL1 is ready (meaning locked).

Note: Before enabling the PLL1, make sure that the reference frequency (F_{ref_ck}) provided to the PLL1 is stable.

The hardware prevents writing PLL1ON to 0 if the PLL1 is currently used to deliver the system clock.

The following PLL1 parameters cannot be changed once the PLL1 is enabled: PLL1M, PLL1SRC, PLL1N, PLL1RGE, PLL1P, PLL1Q, and PLL1R.

To insure an optimal behavior of the PLL1 when one of the postdividers (PLL1P, PLL1Q, or PLL1R) is not used, the application must clear the enable bit (PLL1PEN, PLL1QEN, or

PLL1REN), and, preferably, also the corresponding postdivider field (PLL1P, PLL1Q, or PLL1R).

If the above rules are not respected, the PLL1 output frequency is not ensured.

Output frequency computation

When the PLL1 is operated in integer mode, the VCO frequency (F_{VCO}) is given by the following formula:

$$F_{VCO} = F_{ref_ck} \times PLL1N$$

When the PLL1 is operated in fractional mode, it is possible to change the value of the PLL1FRACN on the fly without disturbing the PLL1 output. This feature can be used either to generate a specific frequency from any crystal value with a good accuracy, or to fine-tune the frequency on the fly.

For PLL1, the VCO frequency is given by the following formula:

$$F_{VCO} = F_{ref_ck} \times (PLL1N + PLL1FRACN / 2^{13})$$

For both integer and fractional mode, the PLL1 output frequency is given by the formula below:

$$F_{pll(y)clk} = F_{VCO} / (PLL1(y) + 1), \text{ with } y = P, Q, \text{ or } R.$$

The PLL1 is disabled by hardware when:

- The system enters Stop or Standby mode.
- A HSE32 failure occurs, and PLL1, clocked by HSE32, is used as the system clock.

The fractional information used by the PLL is reset when disabling the PLL.

PLL1 initialization phase

Here below the recommended PLL1 initialization sequence in integer and fractional mode (PLL1 is supposed to be disabled at the start of the sequence):

1. Initialize the PLL1 registers according to the required frequency.
   - For integer mode, clear PLL1FRACEN.
   - For fractional mode, set PLL1FRACN to the required initial value and then set PLL1FRACEN.
2. Once the PLL1ON bit is set, the application must wait until the PLL1RDY bit is set. As long as PLL1RDY = 0, the PLL1FRACEN bit must not be altered.
3. When the PLL1RDY bit is set, the PLL1 is ready to be used.
4. If the application intends to tune the PLL1 frequency on the fly:
   a) PLL1FRACEN must be cleared so the PLL1FRACN value can be updated while the PLL is running.
   b) A new value can be set in PLL1FRACN.
   c) PLL1FRACEN must be set to activate the newly programmed value in PLL1FRACN and for the PLL to take it into account.

Note: When the PLL1RDY is set, it means that the difference between the PLL1 output frequency and the target value is less than 2 %.

12.4.4 LSE clock

The LSE crystal is a 32.768 kHz low-speed external crystal or ceramic resonator. It has the advantage of providing a low-power but highly accurate clock source to the real-time clock peripheral (RTC) for clock/calendar, the 2.4 GHz RADIO sleep timer, or other timing functions.

The LSE crystal is switched on and off using the LSEON bit in *RCC backup domain control register (RCC_BDCR1)*. If the LSE is used by other peripherals or functions than RTC and TAMP, the LSESYSEN bit must also be set in the *RCC backup domain control register (RCC_BDCR1)* (refer to *LSE when used by peripherals other than RTC/TAMP and RCC functions*).

The crystal oscillator driving strength can be changed at runtime using the LSEDRV[1:0] bits to obtain the best compromise between robustness and a short startup time on the one hand and low-power consumption on the other. The LSE drive can be decreased to a lower-drive capability when the LSE is ON. However, once LSEDRV is selected, the drive capability cannot be increased if LSEON = 1.

The LSERDY flag in the *RCC backup domain control register (RCC_BDCR1)* indicates whether the LSE crystal is stable or not. At startup, the LSE crystal output clock signal is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *RCC clock interrupt enable register (RCC_CIER)*.

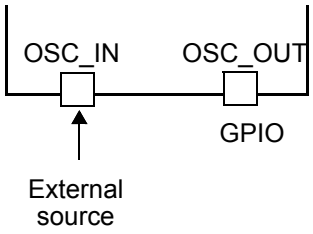
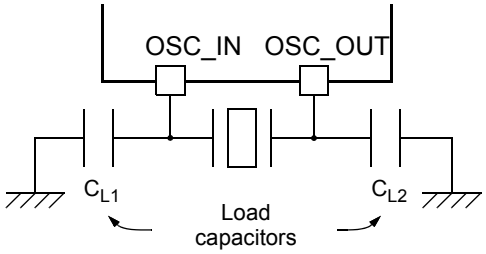
In addition, glitches on the LSE can be filtered by setting LSEGFON. LSEGFON must be written when the LSE is disabled (LSEON = 0 and LSERDY = 0).

The LSE oscillator can be trimmed using the LSETRIM trimming bits in the *RCC backup domain control register (RCC_BDCR1)*. After a BOR0 reset and loading option bytes using the OBL_LAUNCH register bit when the SBF is cleared, the factory trimmed values are loaded in the LSETRIM bits, which can subsequently be modified by the application software.

External source (LSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 1 MHz. This mode is selected by setting the LSEBYP and LSEON bits in the *RCC backup domain control register (RCC_BDCR1)*. The external clock signal (square, sinus, or triangle), with ~50 % duty cycle, must drive the OSC32_IN pin while the OSC32_OUT pin can be used as a GPIO (see *Figure 38*).

Figure 38. LSE 32 clock sources

Clock source	Hardware configuration
External clock	
Crystal/ceramic resonators	

LSE when used by peripherals other than RTC/TAMP and RCC functions

By default, when enabled by LSEON, the LSE is sent only to RTC and TAMP (assuming that RTCSEL = 01).

If the LSE is needed for other peripherals (such as the peripheral clock or a trigger source) or if the LSE is used by an RCC function (such as LSCO or MCO), the lsesys clock must be enabled with LSESYSEN by following the sequence below:

1. Wait until the LSE clock is ready and the LSEON bit is set, so the LSERDY is set in the *RCC backup domain control register (RCC_BDCR1)*.
2. Set the LSESYSEN bit in the RCC_BDCR1 register.
3. Wait until the lsesys clock is ready (LSESYSRDY = 1 in RCC_BDCR1).

The LSE power consumption is increased when LSESYSEN = 1.

12.4.5 LSI clock

The low-power LSI clock can be kept running in Stop and Standby modes for the IWDG, and in Stop 0, 1, 2, and Standby modes for the RTC and TAMP. The LSI clock is generated from two sources

- LSI1 RC low-power oscillator
- LSI2 RC low-drift oscillator selection between LSI1 or LSI2 is done by the LSI2ON bit. Whenever LSI2 is enabled (LSI2ON = 1) and LSI2 is ready (LSY2RDY = 1), the LSI clock is generated by LSI2. Else, LSI1 is selected as the LSI clock source (not available on STM32WBA25/23 devices).

Table 108. LSI clock selection

LSI1ON/LSI1RDY	LSI2ON/LSI2RDY	LSI clock
0/0	0/0	No clock
1/1	0/0	LS1 RC source
0/0	1/1	LSI2 RC source
1/1	1/1	LSI2 RC source

When the IWDG is started, the LSI clock is forced on and cannot be disabled. When both the LSI1 and LSI2 are disabled, LSI1 is forced on. When LSI select LSI2 RC source, the LSI1 RC source can be disabled.

LSI1 low-power

Caution: The LSI1 must not be used for the 2.4 GHz RADIO sleep timer.

The LSI1 RC can be switched on and off using the LSI1ON bit in the *RCC backup domain control register (RCC_BDCR1)*.

The LSI1RDY flag in the *RCC backup domain control register (RCC_BDCR1)* indicates if the LSI1 oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *RCC clock interrupt enable register (RCC_CIER)*.

The clock frequency is either ~32 kHz or ~250 Hz, depending on the LSI1PREDIV bit in the *RCC backup domain control register (RCC_BDCR1)*. Setting LSI1PREDIV results in lower consumption (refer to the electrical characteristics section of the datasheet for more details).

Note: When the IWDG is enabled or when the RTC or TAMP is clocked by the LSI, the LSI1PREDIV can no longer be changed.

LSI2 low-drift (not available on STM32WBA25/23 devices)

The LSI2 RC can be switched on and off using the LSI2ON bit in the RCC Backup domain control register (RCC_BDCR1).

The LSI2RDY flag in the RCC Backup domain control register (RCC_BDCR1) indicates if the LSI2 oscillator is stable or not. At startup, the clock is not released until this bit goes to 1 by hardware. An interrupt can be generated if enabled in the RCC clock interrupt enable register (RCC_CIER). After LSI2 ready, there is an additional delay of up to six clock cycles before clocking a peripheral.

The LSI2 makes possible trimming using the LSI2CFG and LSI2MODE in RCC Backup domain control register (RCC_BDCR2).

- Use LSI2CFG to select the temperature at which the frequency temperature sensitivity is the lowest
- Use LSI2MODE to configure the power consumption versus the accuracy. For best performance nominal power consumption and highest accuracy must be selected.

12.4.6 System clock (SYSCLK) selection

Different clock sources can be used to drive the system clock (SYSCLK):

- HSI16 oscillator
- HSE32 oscillator
- pll1rclk

The system clock maximum frequency is 64 MHz. After a system reset, the HSI16 oscillator at 16 MHz is selected as the system clock. When a clock source is used directly or through the PLL as a system clock, it is not possible to stop it.

A switch from one clock source to another occurs only if the target clock source is ready (fir instance, the clock is stable after a startup delay). If a clock source that is not yet ready is selected, the switch occurs when the clock source is ready. Status bits in the *RCC clock configuration register 1 (RCC_CFGR1)* indicate which clocks are ready and which clock is currently used as the system clock.

Table 109 gives the different bus frequencies depending on the product voltage range.

Table 109. SYSCLK and bus maximum frequency

Product voltage	SYSCLK / AHB1 / AHB2 / AHB4 / AHB5(1) / APB1 / APB2 / APB7	AHB5(2)
range 1	64 MHz	32 MHz
range 1.5		
range 2	16 MHz	12 MHz

1. For PTACONV.
2. For 2.4 GHz RADIO.

Note: After reset, voltage scaling range 2 is used with SYSCLK at 16 MHz, hclk5 at 8 MHz and one wait state on FLASH, SRAM1, and SRAM2.

System clock maximum frequency step

When increasing the SYSCLK frequency, the frequency step must be limited. The frequency must be increased in steps using the PLL1RCLKPRE divider and PLL1RCLKPRESTEP divider steps.

A two-step divider makes it possible to limit the maximum frequency step to 43 MHz, which requires at least 1000 cycles of the pll1rclk clock to switch. Two-step division uses an intermediate frequency of pll1rclk/1.75.

A three-step divider limits the maximum frequency step to 29 MHz and requires at least 2000 cycles of the pll1rclk clock to switch. Three-step division uses intermediate frequencies of pll1rclk/2.25 and pll1rclk/1.4.

As an example, to increase the SYSCLK from 16 MHz (HSI16) to 64 MHz (pll1rclk), the following sequence must be used:

1. Switch VOS to range 1 and wait for VOSRDY.
2. Select PLL1RCLKPRE to divide, and the divider steps in PLL1RCLKPRESTEP.
3. Lock PLL1.
4. Switch the SYSCLK source from HSI16 to PLL1 in SW.
5. Wait for the SYSCLK switch to complete in SWS (SYSCLK = pll1rclk divided).
6. Set PLL1RCLKPRE to not divide.
7. Wait for PLL1RCLKPRERDY to indicate readiness (SYSCLK = pll1rclk).

Warning: The SYSCLK must be switched to a higher frequency when the incremental frequency step is ≤ 47 MHz, or device operation cannot be ensured. For bigger incremental frequency steps, use the PLL1RCLKPRE division.

When the SYSCLK is selected from PLL1 and PLL1RCLKPRE is changed from divide to not divide, the not divide clock is ready on the SYSCLK after a delay, as detailed in *Table 110*.

Table 110. PLL1RCLKPRERDY delay versus pll1rclk frequency

$f_{PLL1RCLK}$ [MHz]	SYSCLK divider delay when going from divide to not divide [µs]	
	Two-step	Three-step
64	15.63	31.25

The PLL1RCLKPRE, PLL1RCLKPRERDY, and SYSCLK frequency behavior depends on the SWS and pll1rclk settings, as detailed in *Table 111*:

Table 111. PLL1RCLKPRERDY and SYSCLK behavior

PLL1RCLKPRE	PLL1ON	PLL1RCLKPRERD	SWS	SYSCLK	Description
0	x	1	Not PLL1	Other than pll1rclk	SWS, select other source then PLL1
1	x	0			
1	1	x	PLL1	pll1rclk	SYSCLK = pll1rclk divided
0	1	0			SYSCLK = pll1rclk divided.
0	1	1			SYSCLK = pll1rclk not divided

System clock frequency change and hclk5

When increasing the SYSCLK frequency above 32 MHz, the hclk5 division ration in HPRE5 must be adapted to keep the hclk5 frequency ≤ 32 MHz.

Warning: The AHB5 clock frequency for the 2.4 GHz RADIO must never exceed 32 MHz. When this is not respected, device operation cannot be ensured.

For this purpose HPRE5 must be written by software with the divider value corresponding to the $f_{PLL1RCLK}$ frequency, before switching the system clock to the PLL1 source in SW. The written HPRE5 value is used by the hardware to divide the SYSCLK, at the same time as the SYSCLK clock switch to the PLL1 source.

1. Lock PLL1.
2. Optional: select PLL1RCLKPRE to divide.
3. Set the HPRE5 value to be used with the $f_{PLL1RCLK}$ frequency.
4. Switch the SYSCLK source to pll1rclk in SW.
5. Wait for the SYSCLK switch to complete in SWS (hclk5 = $f_{PLL1RCLK}$/HPRE5).

When decreasing the SYSCLK frequency from a frequency above 32 MHz from the PLL1 source to the HSE32 or HSE16 source, the hclk5 is set to not divide by hardware.

1. Switch the SYSCLK source away from PLL1.
2. Wait for the SYSCLK switch to complete in SWS (SYSCLK = HSI16 or HSE32, hclk5 = SYSCLK/HDIV5).

The software-written HPRE5 value takes effect only when the SYSCLK has been switched to pll1rclk in SWS.

When SW = PLL1 and SWS = not PLL1, write operations to HPRE5 are ignored.

Note: The HPRE5 divider is not used when the SYSCLK source is HSE32, HSE32 divided by 2, or HSI16 (SWS = not pll1rclk).

In range 2, the hclk5 frequency must be kept at ≤ 8 MHz. Before entering range 2, HDIV5 must select “divide by 2”.

When moving from range 1 or range 1.5 to range 2:

1. Set HDIV5 to divide by 2.
2. Switch the SYSCLK source to HSE32 divided by 2 or HSI16.
3. Wait for the SYSCLK switch to complete in SWS.
4. Select range 2 in VOS in the PWR_VOSR register.
5. Optionally, wait until the ACTVOS in PWR_SVMSR = VOS in PWR_VOSR, and the ACTVOSRDY flag is set in PWR_SVMSR.

When changing from range 2 to range 1 or range 1.5:

1. Select range 1 or range 1.5 in VOS in PWR_VOSR.
2. Wait until the VOSRDY flag is set in PWR_VOSR.
3. Optionally, clear HDIV5 to no longer divide by 2.

Note: When entering Stop 1, 2, and 3 mode, hardware sets the hclk5 frequency to divide by 2 in HDIV5.

Note: The HDIV5 divider is not used when the SYSCLK source is PLL1 (SWS = PLL1).

Note: When 2.4 GHz RADIO is active, the device must be in range 1 or range 1.5, hclk5 must be ≥ 16 MHz, and HDIV5 must be cleared.

12.4.7 Clock source frequency versus voltage scaling

Table 112. Clock source maximum frequency

Product voltage	HSI16	HSE32	PLL1 output (VCO min and max)
Range 1	Allowed	Allowed	64 MHz (128 - 544 MHz)
Range 1.5			
Range 2		Allowed (divided by 2)	Not allowed

12.4.8 HSE32 clock security system (HSECSS)

The HSECSS can be activated by software with the HSECSSON bit. In this case, the clock detector is enabled after the HSE32 oscillator wake-up time and disabled when this oscillator is stopped.

The HSECSS makes it possible to detect the absence of a clock. Refer to the device datasheet for more information.

If a failure is detected on the HSE32 clock, the HSE32 oscillator is automatically disabled. A clock failure event is sent to several timers break input and an interrupt is generated to inform the software about the failure (HSECSSI: clock security system interrupt). This allows the MCU to perform rescue operations. The HSECSSI is linked to the Core NMI (nonmaskable interrupt) exception vector.

Once the HSECSS has been enabled and if the HSE32 clock fails, the HSECSSI occurs and an NMI is automatically generated. The NMI is executed indefinitely, unless the pending HSECSSI bit is cleared. Consequently, the user must clear the HSECSSI bit in the NMI ISR by setting the HSECSSC bit in the *RCC clock interrupt clear register (RCC_CICR)*.

If the HSE32 oscillator is used directly or indirectly as the system clock (indirectly meaning it is used as the PLL1 input clock and the pll1rclk clock is used as the system clock), a detected failure causes a switch of the system clock to the HSI16 oscillator and the disabling of the HSE32 oscillator. If the HSE32 clock and PLL1 are used as system clock when the failure occurs, the PLL1 is disabled too.

12.4.9 LSE clock security system on (LSECSS)

A clock security system on the LSE can be activated by software setting the LSECSSON bit in the *RCC backup domain control register (RCC_BDCR1)*. This bit can be disabled only by a hardware reset or RTC software reset, or after a failure detection on LSE. LSECSSON must be written after LSE is enabled (LSEON set) and ready (LSERDY set by hardware) and after the RTC clock has been selected by RTCSEL.

The LSECSS is working in all modes except Standby.

The clock security system on the LSE detects when the LSE disappears or in case of overfrequency. In addition, the glitches on LSE can be filtered by setting LSEGFON. LSEGFON must be set when LSE is disabled (LSEON = 0 and LSERDY = 0).

If a failure is detected on the external 32 kHz oscillator, the LSE clock is no longer clocking and no hardware action is made to register settings.

In case of an LSECSS detection event (LSECSSD = 1 in the RCC_BDCR1), the software must disable the LSECSSON bit, stop the defective 32 kHz oscillator (disabling LSEON)

and change the low-power clock source (no clock or LSI or HSE32), or take any required action to secure the application.

The LSECSS detection event is connected to the internal tamper 3 of the TAMP peripheral. The internal tamper 3 must be enabled (ITAMP3E = 1 in TAMP_CR1 register) and the associated interrupt enabled (ITAMP3IE in TAMP_IER) to enable wake-up from the low-power modes.

An LSECSS detection event also erases the TAMP backup registers and backup SRAM unless ITAMP3NOER = 1 in the TAMP_CR3 register (see *Section 34: Tamper and backup registers (TAMP)* for more details).

Refer to the product datasheet for LSECSS electrical characteristics.

12.4.10 ADC kernel clock

The ADC kernel clock source is selected with the ADCSEL bit in the *RCC peripherals independent clock configuration register 3 (RCC_CCIPR3)*.

If the application requires that the ADC is precisely triggered by a (LP)TIM without any uncertainty, the hclk must be selected as the ADC kernel clock source. The other clock sources are asynchronous to (LP)TIM, so an uncertainty of the trigger instant is added by the resynchronization between the two clock domains.

12.4.11 RTC and TAMP kernel clock

The RTC kernel clock source is used by RTC and TAMP and can be either the HSE32/32, LSE, or LSI clock. It is selected by programming the RTCSEL bits in the *RCC backup domain control register (RCC_BDCR1)*. This selection cannot be modified without resetting the backup domain. For the RTC to operate properly, the RTC bus clock, pclk, must always be set to a frequency that is greater than or equal to that of the RTC kernel clock.

The TAMP does not require a kernel clock if only the backup registers are used, with tampers in edge detection mode. All other tamper detection modes require a kernel clock (refer to *Section 34: Tamper and backup registers (TAMP)* for more details).

The LSE and LSI clocks are in the backup domain, whereas the HSE32 clock is not. Consequently:

- If LSE or LSI is selected as the RTC or TAMP clock, these peripherals continue to work in Stop 0, 1, 2, and Standby modes, provided the RTCRSB is enabled and V_{DD} supply is maintained.

  The RTC and TAMP are not functional in Stop 3 mode. When RTC and TAMP use LSI or LSE, they are reset when gating off the clock.
- If the HSE32 clock is used as the RTC or TAMP clock, these peripherals work only in Run and Sleep modes. They stop working in Stop and Standby modes. Depending on the TAMP configuration, they can remain functional if used in a mode that does not need a kernel clock.

When the RTC and TAMP clock is LSE or LSI, the RTC and TAMP remain clocked and functional under system reset.

If the LSE is needed only for the RTC or TAMP, LSESYSEN must be kept at reset value to get the lowest consumption.

12.4.12 2.4 GHz RADIO bus clocks

The 2.4 GHz RADIO bus clock can be enabled by software with the RADIOEN and RADIOSMEN bits, and by hardware on a 2.4 GHz RADIO sleep timer wake-up event by the STRADIOCLKON bit. Before accessing the 2.4 GHz RADIO sleep timer registers, the bus clock must be ready, indicated by the RADIOCLKRDY register bit.

Table 113. 2.4 GHz RADIO bus clock control

Device state	CPU state	RADIOEN	RADIOSMEN	STRADIOCLKON	2.4 GHz RADIO state	2.4 GHz RADIO bus clock
X	X	0	X	0	X	Off
Run	RUN	1	X	X	X	On
		X	X	1	X	On
Sleep	SLEEP	1	0	0	X	Off
		1	1	0	ACTIVE	On
					SLEEP/DEEPSLEEP	Off
		X	X	1	X	On
Stop 0	DEEPSLEEP	1	0	0	X	Off
		1	1	0	ACTIVE	On
					SLEEP/DEEPSLEEP	Off
		X	X	1	X	On
Stop 1(1)		X	X	0	SLEEP/DEEPSLEEP	Off
Stop 2(1)(3)					DEEPSLEEP	Off
Stop 3(1)(3)					DEEPSLEEP(2)	Off
Standby(1)(3)	RESET				DEEPSLEEP/POWER DOWN	Off

1. When 2.4 GHz RADIO state is ACTIVE or STRADIOCLKON is 1, the device does not enter Stop 1, Stop 2 Stop 3, and Standby modes.
2. The RADIO sleep timer is also stopped.
3. When 2.4 GHz RADIO state is SLEEP, the device does not enter Stop 2, Stop 3, and Standby modes.

When exiting from a low-power mode and the 2.4 GHz RADIO bus clock has been stopped, the RADIOCLKRDY bit must be rechecked before accessing the 2.4 GHz RADIO registers.

The 2.4 GHz RADIO bus clock is kept active only in low-power modes, when STRADIOCLKON is set and/or RADIOEN and RADIOSMEN are set, and the 2.4 GHz RADIO is active.

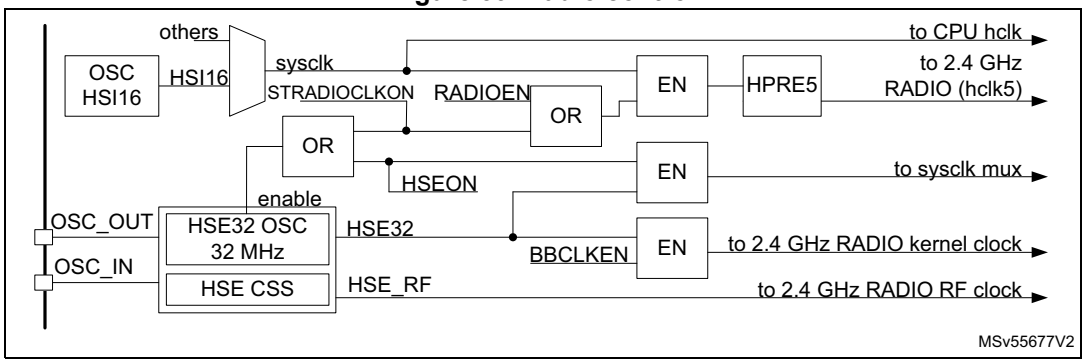
12.4.13 2.4 GHz RADIO kernel clocks

The 2.4 GHz RADIO has different kernel clocks:

- Baseband kernel clock
- Sleep timer low-power clock

The 2.4 GHz RADIO baseband kernel clock is enabled by the BBCLKEN register bit. This clock has the HSE32 as its clock source. For this purpose, the HSE32 oscillator is enabled by hardware on a 2.4 GHz RADIO sleep timer wake-up event setting the STRADIOCLKON register bit, or by software setting the HSEON register bit. When the 2.4 GHz RADIO no longer needs the HSE32 and bus clocks, software must clear the BBCLKEN and STRADIOCLKON bits. If HSE32 is not used by any other function, it must also clear the HSEON bits.

Figure 39. Radio control



In range 2, the 2.4 GHz RADIO baseband kernel clock is divided by 4 by the hardware. The 2.4 GHz RADIO baseband kernel clock must be enabled to access the 2.4 GHz RADIO RXTX RAM or sequence RAM.

Moreover, the 2.4 GHz RADIO bus clock (hclk5) is enabled by hardware through the STRADIOCLKON bit, independently from the RADIOEN and RADIOSMEN bits. When the 2.4 GHz RADIO no longer needs its bus clock, software must clear the STRADIOCLKON bit. The STRADIOCLKON bit also keeps the 2.4 GHz RADIO bus clock and baseband kernel clock active when entering low-power modes.

Outside any scheduled radio activity, when in Run mode, software can enable the 2.4 GHz RADIO bus clock through the RADIOEN and RADIOSMEN register bits. In this case, the RADIOSMEN bit keeps the 2.4 GHz RADIO bus clock and baseband kernel clock active when entering low-power modes.

The 2.4 GHz RADIO sleep timer kernel clock source can be either the HSE32/1000, LSI, or LSE clock. LSI must be used only when its source is LSI2. It is selected by setting the RADIOSTSEL bits in the *RCC backup domain control register (RCC_BDCR1)*.

12.4.14 Timer kernel clock

The timer (TIM) kernel clock frequency is derived from the bus clock pclk. The frequency is automatically defined by hardware:

- If the APB prescaler equals 1, the timer clock frequencies are set to the APB domain frequency (kernel clock frequency = pclk frequency)
- If the APB prescaler does not equal 1, he timer clock frequencies are set to twice (×2) the APB domain frequency (kernel clock frequency = 2 x pclk frequency)

12.4.15 Independent watchdog kernel clock

The independent watchdog uses the LSI as the kernel clock.

If the independent watchdog (IWDG) is started by either user option bytes or software and the LSI clock is disabled (LSI1ON and LSI2ON are cleared), the LSI1 oscillator is forced on. After the LSI oscillator ready delay, the LSI clock is provided to the IWDG.

12.4.16 USB kernel clock

The USB uses the pll1pclk at 48 MHz as the kernel clock. Both the HSE32 and PLL1 shall be enabled.

12.4.17 XSPI feedback clock delay

The XSPI1 feedback clock delay is performed by the DLYBXS1 delay block. It allows for a delay to be added to the XSPI1 feedback clock (refer to *Section 21: Delay block (DLYB)*).

12.4.18 SysTick calibration value register

The Cortex®-M33 core with the TrustZone® security extension embeds two SysTick timers.

When TrustZone® is activated, the following SysTick timers are available:

- SysTick, secure instance
- SysTick, nonsecure instance

When TrustZone® is disabled, only one SysTick timer is available.

The Cortex®-M33 SysTick timer calibration value (STCALIB) is 0x3E8. It gives a reference time base of 1 ms based on a SysTick clock frequency of 1 MHz. To match the 1 ms time base for an application running at a given frequency, the SysTick reload value must be programmed as follows in the Cortex®-M33 SYST_RVR register:

- When the SysTick clock source is the CPU clock hclk1
  Reload value = (f_{HCLK1} x STCALIB) - 1
- When the SysTick clock source is an external clock (hclk1 divided by 8)
  Reload value = ((f_{HCLK1}/8) x STCALIB) - 1

Example: the SysTick clock source is the CPU clock hclk1 at 100 MHz, to match a time base of 1 ms:

SysTick reload value = (100 x STCALIB) - 1 = 0x0001 869F

Note: When using debug Stop mode (DBG_STOP), before the CPU enters Deepsleep it is considered good practice to have software disable the SysTick.

12.4.19 Clock-out capability

- **MCO**

  The microcontroller clock output (MCO) capability allows the clock to be output onto the external MCO pin. One of the following clock signals can be selected as the MCO clock:

  – Active in Run, Sleep, and Stop 0, 1, and 2 modes:

  - LSI

  - LSE

  - HSI16 (in Stop 0, 1, and 2 modes only when an autonomous peripheral kernel clock request is active, or when enabled by HSIKERON)

  - HSE32 (in Stop 0 mode only when the 2.4 GHz RADIO kernel clock request is active)

  - SYSCLKpre (in Stop 0 mode when an autonomous peripheral, other than the 2.4 GHz RADIO, bus clock request is active)

  - hclk5 (when enabled by RCC_AHB5CR.RADIOEN, and in Stop 0 mode when the 2.4 GHz RADIO kernel clock request is active)

  – Active only in Run and Sleep modes:

  - pll1pclk

  - pll1qclk

  - pll1rclk

  The selection is controlled by the MCOSEL bits in the *RCC clock configuration register 1 (RCC_CFGR1)*. The selected clock can be divided with the MCOPRE field in the *RCC clock configuration register 1 (RCC_CFGR1)*.

  The MCO clock output requires the corresponding GPIO pin alternate function to select MCO.

- **LSCO**

  Slow clock output (LSCO) allows one of the low-speed clocks to be output onto the external LSCO pin:

  – LSI

  – LSE

  This output remains available in all Run, Sleep, Stop, and Standby modes. The selection is controlled by the LSCOSEL bit and enabled with the LSCOEN bit in the *RCC backup domain control register (RCC_BDCR1)*.

12.4.20 Internal/external clock measurement

The HSI16 and LSI frequency can be measured indirectly by the TIM16 or TIM17 channel 1 input capture and LPTIM1 or LPTIM2 channel 2 input capture.

HSI16 calibration using LSE

The primary purpose of connecting the LSE to the channel 1 input capture of TIM16 and TIM17 to the channel 2 input capture of LPTIM1 is to be able to measure the HSI16 frequency precisely. When using TIM16 or TIM17 for this purpose, the HSI16 must be used as the system clock source.

The number of HSI16 clock counts between two edges of the LSE signal provides a measure of the internal clock period. Taking advantage of the high precision of the LSE

crystal (typically a few hundred ppm), the internal clock frequency can be determined with a similar resolution, depending on the measurement time. The HSI16 can be trimmed to compensate for the process, temperature, and/or voltage-related frequency deviations.

The basic concept consists of providing a relative measurement (such as the HSI16/LSE ratio). The precision is therefore closely related to the ratio between the two clock sources: the higher the ratio, the better the measurement.

The HSI16 oscillator has dedicated user-accessible calibration bits (HSITRIM) for this purpose.

HSI16 calibration using HSE32

The HSE32 must be used as the system clock and the timer input capture must be connected to HSI16/256. TIM16 and TIM17 channel 1 input capture, as well as the LPTIM2 input capture 2, are connected to the divided oscillator only when TIMICSEL is set in the *RCC peripherals independent clock configuration register 1 (RCC_CCIPR1)*.

LSI calibration using HSE32

The calibration of the LSI follows the same principle as the HSI16 calibration, but with a change of the reference clock. The LSI clock must be connected to the channel 1 input capture of TIM16 or TIM17, or to the channel 2 input capture of LPTIM1. Then, HSE32 must be defined as the system clock source. The number of HSE32 clock counts between edges of the LSI signal provides a measure of the internal low-speed clock period.

The basic concept consists of providing a relative measurement (such as the HSE32/LSI ratio). The precision is therefore closely related to the ratio between the two clock sources: the higher the ratio, the better the measurement.

12.4.21 Audio synchronization

The audio synchronization system is used to provide capture compare information between the Bluetooth® LE radio packet timing and audio clock.

- 20-bit programmable free-running upcounter
- Autoreload
- Clock prescaler
- Compare
- Input capture
- Capture period
- Interrupt:
  - Input capture
  - Compare
  - Compare error
  - Autoreload

Before enabling the audio synchronization counter, the autoreload, clock prescaler, and capture prescaler must be provided.

- The clock prescaler is used to provide a lower-speed clock to the counter.
- Autoreload is used to define the audio synchronization counter period.
- The capture prescaler is used to define the capture period. The capture period is a multiple of the audio synchronization counter period.

A capture value of the counter and the capture prescaler value are updated on the first audio synchronization trigger event in the capture period. Any subsequent audio trigger synchronization events during this capture period are discarded. When enabled, an associated capture event interrupt can be generated.

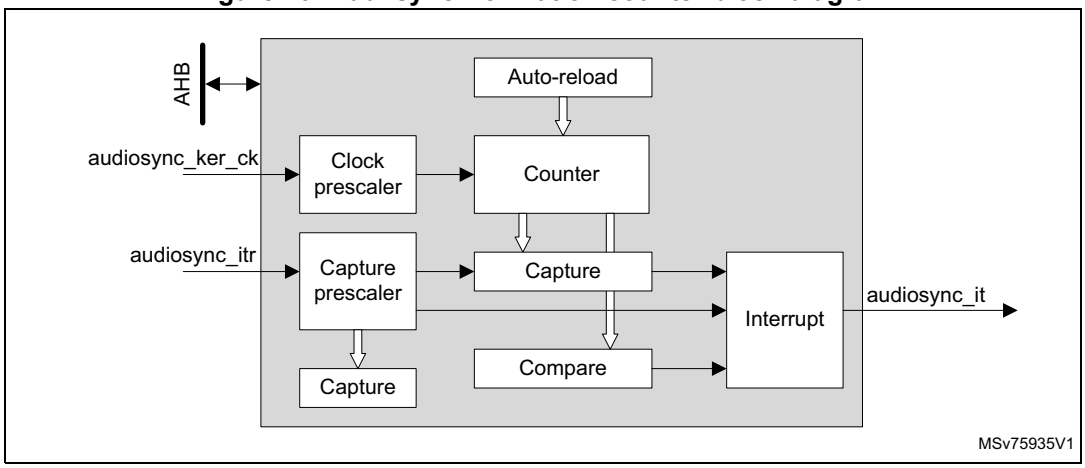
After enabling the audio synchronization counter in CEN, the capture prescaler only starts counting the capture period after receiving a first synchronization from a Bluetooth® LE radio packet.

A capture error flag is set when no audio trigger synchronization event occurs during the capture period. When enabled, an associated capture error interrupt event can be generated.

The compare can be used to generate an interrupt when the counter reaches the compare value.

Refer to *Figure 40: Audi synchronization counter block diagram* below for the block diagram.

Figure 40. Audi synchronization counter block diagram

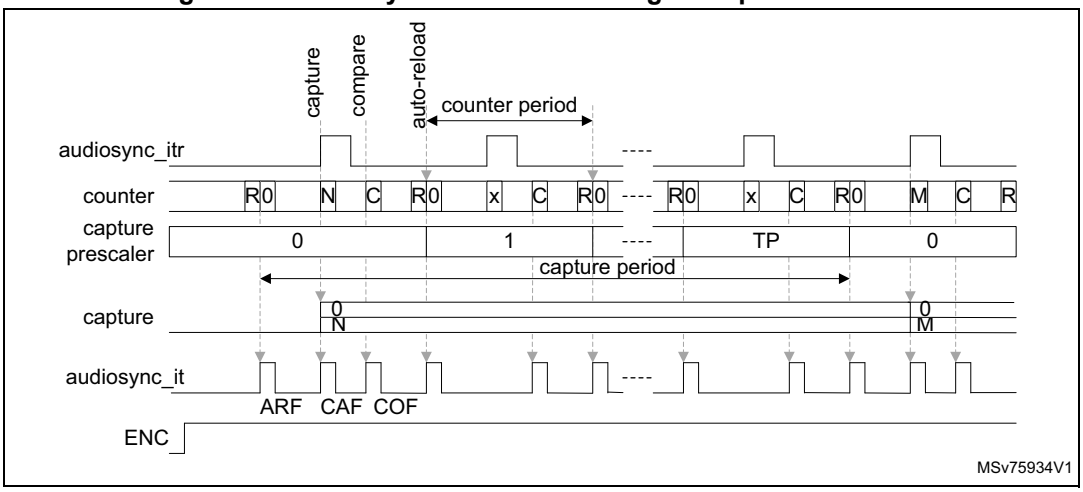


How to use the audio synchronization counter

In the example in *Figure 41* below, the following parameters have been used:

- Autoreload = R (counter period)
- Capture prescaler = TP (capture period)
- Compare = C

Figure 41. Audio synchronization timing example



The capture value is updated only for the first audiosync_itr event in the capture period. In the first capture period, counter value N and capture prescaler value 0 are captured. The capture period counter is only started after the very first audiosync_itr event, after enabling the audio synchronization counter with ENC. The other audiosync_itr events during this capture period are discarded. In the next capture period, counter value M and capture prescaler value 0 are captured.

The counter drift is calculated by software as follows:

$$Drift = 10^6 x(N-M) / (RxTP) \quad (ppm)$$

The trigger event may be delayed due to missing received packets by the 2.4 GHz RADIO. The calculation must compensate for this delay.

Error handling

When no audiosync_itr event has been received during the capture period, a capture error interrupt is generated when enabled.

12.4.22 Peripheral clock gating and autonomous mode

Peripheral clock gating in Run mode

Each peripheral clock can be enabled by setting the corresponding EN bit in the RCC_AHBxENR and RCC_APBxENR registers.

When the peripheral clock is not active, read or write access operations to the peripheral registers are not supported.

The enable bit has a synchronization mechanism to create a glitch-free clock for the peripheral. After the enable bit is set, the clock is active after two cycles of the peripheral bus clock.

Caution: Just after enabling the peripheral clock, the software must wait for these two clock cycles before accessing the peripheral registers.

Peripherals clock gating in Sleep and Stop modes

When a peripheral is enabled in RCC_AHBxENR or RCC_APBxENR registers, its bus and kernel clocks can be automatically gated off when the device is in Sleep and Stop modes, by clearing the peripheral SMEN bit in the RCC_AHBxSMENR or RCC_APBxSMENR register. Both the EN and SMEN bits of the peripheral must be set to keep the peripheral bus and kernel clocks on in Sleep and Stop modes. Moreover, in Stop modes, the peripheral bus and kernel clocks are active only upon peripheral clock requests. This excludes the 2.4 GHz RADIO, which requests its bus clock independently from the setting in the RADIOEN and RADIOSMEN register bits.

The peripheral bus clock runs only in Sleep and Stop 0 modes.

The peripheral kernel clock runs only in Sleep and Stop 0, 1, and 2 modes.

For the 2.4 GHz RADIO, the bus clock runs only in Sleep and Stop 0 modes when the STRADIOCLKON bit is set, or when the 2.4 GHz RADIO is active and RADIOEN and RADIOSMEN are set.

Caution: All peripherals on the same bus, with the SMEN bit set, get a clock when an autonomous peripheral on this same bus requests its clock. Peripherals that are not supposed to be clocked in Stop mode must have their SMEN bit cleared.

Caution: The SMEN bit of the peripheral must be set to allow the generation of an interrupt capable of waking up the device from Sleep and Stop modes. This is not necessary when the peripheral wake-up interrupt is generated through the EXTI (GPIO and PVD).

Peripheral clock gating and autonomous mode in Stop modes

Some peripherals support autonomous mode (refer to *Table 114: Autonomous peripherals*). These peripherals can generate a kernel clock request and an AHB/APB bus clock request when they need it, to operate and update their status register even in Stop 0, 1, and 2 modes. Depending on the peripheral configuration, either a DMA request or an interrupt can be associated with the peripheral event.

When the system enters a low-power mode (Stop or Standby) and an autonomous peripheral bus clock request is active, or upon an autonomous peripheral bus clock request during Stop mode, Stop 0 mode is entered and the HSI16 oscillator is kept active or woken up and selected as the SYSCLK. The bus clocks for all peripherals having their clock enabled through the EN and SMEN bits are activated.

Note: As soon as the CPU enters Deepsleep, the system enters Stop mode and the autonomous mode operation peripheral bus clock and SYSCLK is switched to HSI16 at 16 MHz. If autonomous peripheral operation with higher bus clock frequencies is needed, the CPU must enter Sleep mode and keep the system in Run with the configured Run mode SYSCLK clock frequency.

If the autonomous peripheral is configured with DMA requests enabled, a data transfer is performed using the peripheral bus clock. The bus clocks and the HSI16 oscillator are switched off automatically when the transfer is finished and no other peripheral requests its bus clock or HSI16 derived kernel clock.

If the autonomous peripheral is configured with interrupt enabled, the interrupt wakes up the device into Run mode.

The autonomous peripherals are autonomous in Stop 0 mode with the LPDMA1, SRAM1, and SRAM2, and are autonomous in Stop 1 and Stop 2 modes on their kernel clock.

There is no autonomous peripheral operation in Stop 3 mode.

Table 114 shows the list of peripherals with autonomous mode capability.

Table 114. Autonomous peripherals

<table>
<tr><th rowspan="2">Domain</th><th rowspan="2">Peripheral</th><th colspan="4">Autonomous(1)</th><th rowspan="2">Associated DMA(2)</th><th rowspan="2">Associated SRAM(2)</th></tr>
<tr><th>Stop 0</th><th>Stop 1</th><th>Stop 2</th><th>Stop 3</th></tr>
<tr><td rowspan="3">AHB1, APB1, APB2</td><td>U(S)ART1</td><td rowspan="8">Yes</td><td colspan="2" rowspan="3">No</td><td rowspan="8">LPDMA1</td><td rowspan="8">SRAM1, SRAM2</td></tr>
<tr><td>I2C1</td></tr>
<tr><td>LPTIM2</td></tr>
<tr><td rowspan="5">AHB4, APB7</td><td>LPUART1</td><td rowspan="4">Yes (3)</td><td rowspan="5">No</td></tr>
<tr><td>SPI3</td></tr>
<tr><td>I2C3</td></tr>
<tr><td>LPTIM1</td></tr>
<tr><td>ADC4</td><td>No</td></tr>
<tr><td>AHB5</td><td>2.4 GHz RADIO + RXTXRAM</td><td>Yes (4)</td><td colspan="3">No</td><td>-</td><td>-</td></tr>
</table>

1. Enabled when both the EN and SMEN bits of the peripheral are set.
2. DMA and SRAM are only usable in Stop 0 and Stop 1 modes.
3. Without DMA.
4. Enabled when the 2.4 GHz RADIO STRADIOCLKON and BBCLKEN bits are set and the 2.4 GHz RADIO is active. Available only in Stop 0 range 1 and range 1.5.

For peripherals, the autonomous mode is enabled in Stop modes if both the EN and SMEN bits of the peripheral are set.

For the 2.4 GHz RADIO sleep timer it is operational in Stop 0, 1, and 2 modes and Standby with retention. Waking up from the sleep timer puts the system in Run mode and enables the 2.4 GHz RADIO bus clock. The 2.4 GHz RADIO active mode is enabled by software when, additionally, BBCLKEN bit is set, and allows autonomous operation in Stop 0 range 1 and range 1.5 mode.

If an autonomous peripheral requests its kernel clock in Stop 0, 1, and 2 modes, the internal oscillator (HSI16) is woken up if it was off and the kernel clock is propagated only to the peripheral requesting it, and the EN and SMEN bits of the peripheral are set. When the peripheral releases its kernel clock request, the HSI16 is switched off if no other peripheral requests it. Only the 2.4 GHz RADIO uses the HSE32 as the kernel clock, which is woken up, if it was off, by a 2.4 GHZ RADIO each time it is woken up by the sleep timer.

If an autonomous peripheral requests its bus clock in Stop 0, 1, and 2 modes and the EN and SMEN bits of the peripheral are set, the internal oscillator (HSI16) is woken up, if it was off, and the system clock is propagated to all peripherals on the associated AHB bus configured with both the EN and SMEN bits of the peripheral set.

Caution: The bus clock propagates to all peripherals (autonomous and nonautonomous peripherals) on the same AHB bus that have both the EN SMEN bits set.

HSI16 can be forced to remain on in Stop 0, 1, and 2 modes by configuring HSIKERON in the RCC_CR register. In this case, the oscillator is propagated only to the peripheral kernel clocks of the enabled autonomous peripherals, which select this oscillator as the kernel clock. This allows the peripheral baud or conversion rates to increase, as there is no need to wait for the oscillator wake-up time when the peripheral requests its kernel clock.

The LSE or LSI always remains on in Stop modes and can be used as the kernel clock in Stop 0, 1, and 2 modes.

12.5 RCC security and privilege functional description

12.5.1 RCC TrustZone® security protection modes

TrustZone® security is activated by the TZEN user option bit in the FLASH_OPTR register. The RCC can secure RCC configuration and status bits from being modified by nonsecure access operations through the *RCC secure configuration register (RCC_SECCFGR)*. This prevents nonsecure access to reading or modifying the following features:

- HSE32, HSECSS, HSI16, LSI, LSE, LSECSS, and LSCO configuration and status bits
- PLL1, AHB, and APB prescaler configuration and status bits
- System clock (SYSCLK) source clock selection and status bits
- MCO clock output configuration bits
- Remove reset flag RMVF configuration

If SPRIV is set in the *RCC privilege configuration register (RCC_PRIVCFGR)*, the RCC_SECCFGR register can be written only by secure and privileged access. If SPRIV is cleared in RCC_PRIVCFGR, RCC_SECCFGR can be written only through secure access, privileged or unprivileged.

RCC_SECCFGR can be read through secure, nonsecure, privileged, and unprivileged access.

When a peripheral is configured as secure, its related clock, reset, clock source selection, and clock enable during low-power modes control bits are also secure (refer to *Table 115: RCC security configuration summary*).

A peripheral is secure when:

- For securable peripherals by GTZC1_TZSC (TrustZone® security controller): the SEC security bit in the secure configuration registers corresponding to this peripheral is set.
- For TrustZone®-aware peripherals: a security feature of this peripheral is enabled through its dedicated bits.

Table 115 summarizes the RCC secured bits following the security configuration bit in the RCC_SECCFGR register.

When one security configuration bit is set, some configuration and status bits are secured. The RCC registers may contain secure and nonsecure bits:

- Secured bits: read and write operations are allowed only by secure access. Nonsecure read returns 0 and write access operations are ignored. No illegal access event is generated.
- Nonsecure bits: no restriction. Read and write operations are allowed by both secure and nonsecure access operations.
- A nonsecure write access to RCC_SECCFGR is ignored and generates an illegal access event. An illegal access interrupt is generated if the RCC illegal access interrupt is enabled in the GTZC1_TZIC registers. RCC_SECCFGR can be read by secure or nonsecure access.

When the TrustZone® security is disabled (TZEN = 0), all registers are nonsecure. RCC_SECCFGR write access operations are ignored.

Table 115. RCC security configuration summary

Configuration bit in RCC_SECCFGR	Secured bits	Corresponding register
HSISEC	HSION, HSIKERON, HSIRDY	RCC_CR
	HSICAL, HSITRIM	RCC_ICSCR3
	HSIRDYIE	RCC_CIER
	HSIRDYIF	RCC_CIFR
	HSIRDYC	RCC_CICR
HSESEC	HSEON, HSERDY, HSECSSON, HSEPRE	RCC_CR
	HSERDYIE	RCC_CIER
	HSERDYIF, HSECSSF	RCC_CIFR
	HSERDYC, HSECSSC	RCC_CICR
	HSETRIM	RCC_ECSCR1
LSISEC	LSI1ON, LSI1RDY, LSI1PREDIV, LSI2ON, LSI2RDY, LSCOSEL, LSCOEN	RCC_BDCR1
	LSI2MODE, LSI2CFG	RCC_BDCR2
	LSI1RDYIE	RCC_CIER
	LSI1RDYIF	RCC_CIFR
	LSI1RDYC	RCC_CICR
LSESEC	LSECSSON, LSECSSD, LSEDRV, LSEBYP, LSERDY, LSEON, LSEGFON, LSESYSRDY, LSESYSEN, LSCOSEL, LSCOEN	RCC_BDCR1
	LSERDYIE, LSI2RDYIE	RCC_CIER
	LSERDYF, LSI2RDYF	RCC_CIFR
	LSERDYC, LSI2RDYC	RCC_CICR
SYSCLKSEC	SW, SWS, MCOSEL, MCOPRE	RCC_CFGR1
	SYSTICKSEL	RCC_CCIPR1
	VOS	PWR_VOSR

Table 115. RCC security configuration summary (continued)

Configuration bit in RCC_SECCFGR	Secured bits	Corresponding register
PRESCSEC	HPRE, PPRE1, PPRE2	RCC_CFGR2
	PPRE7	RCC_CFGR3
	HPRE5, HDIV5	RCC_CFGR4
PLL1SEC	PLL1SRC, PLL1RGE, PLL1FRACEN, PLL1M, PLL1PEN, PLL1QEN, PLL1REN, PLL1RCLKPRE, PLL1RCLKSTEP, PLL1RCLKPRERDY	RCC_PLL1CFGR
	PLL1N, PLL1P, PLL1Q, PLL1R	RCC_PLL1DIVR
	PLL1FRACN	RCC_PLL1FRACR
	PLL1RDY, PLL1ON	RCC_CR
	PLL1RDYIE	RCC_CIER
	PLL1RDYF	RCC_CIFR
	PLL1RDYC	RCC_CICR
RMVFSEC	RMVF	RCC_CSR

12.5.2 RCC privilege protection modes

By default, after reset, all RCC registers can be read or written with both privileged and unprivileged access, except the *RCC privilege configuration register (RCC_PRIVCFGR)*, which can be written only with privileged access. RCC_PRIVCFGR can be read by secure and nonsecure, privileged and unprivileged access.

The SPRIV bit in RCC_PRIVCFGR can be written only with secure privileged access. This bit configures the privileged access of all RCC secure functions (as defined by the *RCC secure configuration register (RCC_SECCFGR)* or by the GTZC1_TZSC for securable peripherals, or by the peripheral itself in the case of TrustZone®-aware peripherals).

When the SPRIV bit is set in RCC_PRIVCFGR:

- Writing the RCC secure bits is possible only with privileged access, including RCC_SECCFGR.
- The RCC secure bits can be read only with privileged access except RCC_SECCFGR and RCC_PRIVCFGR that can be read by privileged or unprivileged access.
- An unprivileged access to a privileged RCC bit or register is discarded: the bits are read as zero and the write to these bits is ignored (RAZ/WI).

The NSPRIV bit in RCC_PRIVCFGR can be written with privileged access only, secure or nonsecure. This bit configures the privileged access of all nonsecure RCC functions (as defined by RCC_SECCFGR, the GTZC1_TZSC for securable peripherals, or the peripheral itself for TrustZone®-aware peripherals).

When the NSPRIV bit is set in RCC_PRIVCFGR:

- Writing the nonsecure RCC bits is possible only with privileged access.
- The nonsecure RCC bits can be read only with privileged access, except RCC_PRIVCFGR, which can be read with privileged or unprivileged access.
- An unprivileged access operation to a privileged RCC bit or register is discarded: the bits are read as zero and the write to these bits is ignored (RAZ/WI).

12.6 RCC low-power modes

- AHB and APB peripheral clocks, including the DMA clock, can be disabled by software.
- Sleep mode stops the CPU hclk1 clock. The memory interface clocks (flash memory, cache, and all SRAM interfaces) can be stopped by software during Sleep mode. The AHB to APB bridge clocks are disabled by hardware during Sleep mode, when all clocks of the peripherals connected to them are disabled.
- Stop modes stop all clocks in the core domain and disable the PLL1, HSI16, and HSE32 oscillators. However, HSI16 and HSE32 can be switched on if the peripheral requests it for kernel clock operation purposes, or to generate a wake-up interrupt (see *Section 12.4.22: Peripheral clock gating and autonomous mode* for more details). LSI and LSE can also remain active in Stop modes.
- Standby mode stops all the clocks and disables the PLL1, HSI16, and HSE32 oscillators. LSI and LSE can remain active in Standby modes.

Stopping the system clock in Stop and Standby modes can be overridden for debugging by setting the DBG_STOP and/or DBG_STANDBY bits. For more details, refer to *Section 41.13.4: Low-power mode emulation*.

Note: When using debug Stop mode (DBG_STOP), a SysTick event wakes up the device. It is considered good practice that software disables the SysTick before the CPU enters Deepsleep.

When entering and exiting Stop modes, the system clock is HSI16. The user trim of HSI16 is kept.

When leaving the Standby modes, the system clock is HSI16. The user trim is lost.

If a flash memory programming operation is in progress, Stop and Standby mode entry is delayed until the flash memory interface access operation has finished. If access to the APB domain is ongoing, Stop and Standby mode entry is delayed until the APB access operation has finished. If an autonomous peripheral bus clock request is active, Stop 0 mode is entered. If another low-power mode (Stop or Standby) is selected in LPMS, entry to the selected low-power mode is delayed until the autonomous peripheral bus clock request is released.

12.7 RCC interrupts

Table 116 summarizes the interrupt sources and the way to control them.

Table 116. Interrupt sources and control

Interrupt vector	Interrupt event flag	Description	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop, Standby modes
RCC/RCC_S(1)	LSI1RDYF	LSI1 ready	LSI1RDYIE	Set LSI1RDYC to 1	Yes	No
	LSI2RDYF	LSI2 ready	LSI2RDYIE	Set LSI2RDYC to 1		
	LSERDYF	LSE ready	LSERDYIE	Set LSERDYC to 1		
	HSIRDYF	HSI16 ready	HSIRDYIE	Set HSIRDYC to 1		
	HSERDYF	HSE32 ready	HSERDYIE	Set HSERDYC to 1		
	PLL1RDYF	PLL1 ready	PLL1RDYIE	Set PLL1RDYC to 1		
TAMP	ITAMP3F(2)	LSECSS failure	LSECSSON and ITAMP3E(2) and ITAMP3IE(2)	Set CITAMP3F(2) to 1	Yes	Yes
NMI	HSECSSF	HSECSS failure	HSECSSON(3)	Set HSECSSC to 1	Yes	No
AUDIOSYNC	CAF	Capture event	CAIE	Write CAF to 0	Yes	No
	COF	Compare event	COIE	Write COF to 0		
	CAEF	Capture error event	CAEIE	Write CAEF to 0		
	ARF	Auto-reload event	ARIE	Write ARF to 0		

1. The RCC secure interrupt vector is used only when TrustZone® is enabled.
2. The LSECSS failure event (LSECSSD) is connected to the TAMP internal tamper 3. To get the interrupt associated with this event, the internal tamper 3 and its interrupt must be enabled. The ITAMP3F, ITAMP3E, ITAMP3IE, and CITAMP3F bits are in the TAMP peripheral.
3. It is not possible to mask this interrupt when the security system feature is enabled (HSECSSON = 1).

12.8 RCC registers

12.8.1 RCC clock control register (RCC_CR)

Address offset: 0x000

Reset value: 0x0000 0500

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	PLL1RDY	PLL1ON	Res.	Res.	Res.	HSEPRE	HSECSSON	Res.	HSERDY	HSEON
						r	rw				rw	rs		r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSIRDY	HSIKERON	HSION	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					r	rw	rw								

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **PLL1RDY**: PLL1 clock ready flag

Set by hardware to indicate that the PLL1 is locked.

Access to the bit can be secured by RCC PLL1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: PLL1 unlocked

1: PLL1 locked (PLL1RDY remains set when PLL1 is selected as sysclk and PLL1 is disabled by HSECSS failure).

Bit 24 **PLL1ON**: PLL1 enable

Set and cleared by software to enable the main PLL.

Cleared by hardware when entering Stop or Standby modes and when PLL1 on HSE32 is selected as sysclk, on a HSECSS failure.

Access to the bit can be secured by RCC PLL1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: PLL1 off

1: PLL1 on

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **HSEPRE**: HSE32 clock for SYSCLK prescaler

Set and cleared by software to control the division factor of the HSE32 clock for SYSCLK.

Access to the bit can be secured by RCC HSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: HSE32 not divided, SYSCLK = HSE32

1: HSE32 divided, SYSCLK = HSE32/2

Bit 19 **HSECSSON**: HSE32 clock security system enable

Set by software to enable the HSE32 clock security system. When HSECSSON is set, the clock detector is enabled by hardware when the HSE32 oscillator is ready and disabled by hardware if a HSE32 clock failure is detected. This bit is set only and is cleared by reset.

Access to the bit can be secured by RCC HSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: HSE32 clock security system off (clock detector off)

1: HSE32 clock security system on (clock detector on if the HSE32 oscillator is stable, off if not).

Bit 18 Reserved, must be kept at reset value.

Bit 17 **HSERDY**: HSE32 clock ready flag

Set by hardware to indicate that the HSE32 oscillator is stable. This bit is set both when HSE32 is enabled by software by setting HSEON and when requested as kernel clock by the 2.4 GHz RADIO.

Access to the bit can be secured by RCC HSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: HSE32 oscillator not ready

1: HSE32 oscillator ready to be used by the CPU.

Bit 16 **HSEON**: HSE32 clock enable

Set and cleared by software.

Cleared by hardware to stop the HSE32 clock for the CPU when entering Stop and Standby modes and on a HSECSS failure.

When the HSE32 is used as a 2.4 GHz RADIO kernel clock, enabled by RADIOEN and RADIOSMEN, and the 2.4 GHz RADIO is active, HSEON is not cleared when entering a low-power mode. In this case, only Stop 0 mode is entered as a low-power mode.

This bit cannot be reset if the HSE32 oscillator is used directly or indirectly as the system clock.

Access to the bit can be secured by RCC HSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: HSE32 oscillator not requested by the CPU.

1: HSE32 oscillator ON

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **HSIRDY**: HSI16 clock ready flag

Set by hardware to indicate that HSI16 oscillator is stable. This bit is set only when HSI16 is enabled by software by setting HSION.

Access to the bit can be secured by RCC HSISEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: HSI16 clock not ready

1: HSI16 clock ready

Note: Once the HSION bit is cleared, HSIRDY goes low after six HSI16 clock cycles.

Bit 9 **HSIKERON**: HSI16 enable for some peripheral kernels

Set and cleared by software to force the HSI16 oscillator on, even in Stop 0, 1, and 2 modes. Keeping the HSI16 oscillator on in Stop modes allows the communication speed not to be reduced by the HSI16 oscillator startup time. This bit has no effect on the HSION register bit value.

Cleared by hardware when entering Stop 3 and Standby modes.

Refer to *Peripheral clock gating and autonomous mode* for more details.

Access to the bit can be secured by RCC HSISEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: No effect on HSI16 oscillator

1: HSI16 oscillator forced on even in Stop 0, 1, and 2 modes

Bit 8 **HSION**: HSI16 clock enable

Set and cleared by software.

Cleared by hardware when entering Stop and Standby modes.

Set by hardware to force the HSI16 oscillator on when exiting Stop and Standby modes.

Set by hardware to force the HSI16 oscillator on in case of clock security failure of the HSE32 crystal oscillator.

This bit is set by hardware if the HSI16 is used directly or indirectly as the system clock.

Access to the bit can be secured by RCC HSISEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: HSI16 oscillator off

1: HSI16 oscillator on

Bits 7:0 Reserved, must be kept at reset value.

12.8.2 RCC internal clock sources calibration register 3 (RCC_ICSCR3)

Address offset: 0x010

Reset value: 0x0010 0XXX

X is factory-programmed.

Access: no wait state; word, half-word, and byte access

Access to this register can be protected by RCC HSISEC and RCC SPRIV or RCC NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSITRIM[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSICAL[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **HSITRIM[4:0]**: HSI16 oscillator trimming

These bits provide an additional user-programmable trimming value that is added to the HSICAL[11:0] bits. It can be programmed to adjust to voltage and temperature variations that influence the frequency of the HSI16.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSICAL[11:0]**: HSI16 oscillator calibration
These bits are initialized at startup with the factory-programmed HSI16 calibration value.

12.8.3 RCC clock configuration register 1 (RCC_CFGR1)

Address offset: 0x01C

Reset value: 0x0000 0000

Access: 0 ≤ wait state ≤ 2; word, half-word, and byte access

One or two wait states are inserted only if access occurs during a clock source switch.

Access to this register can be protected by RCC SYSCLKSEC and RCC SPRIV, or RCC NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MCOPRE[2:0]			MCOSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWS[1:0]		SW[1:0]	
												r	r	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **MCOPRE[2:0]**: microcontroller clock output prescaler
Set and cleared by software.
It is highly recommended to change this prescaler before MCO output is enabled.
000: MCO divided by 1
001: MCO divided by 2
010: MCO divided by 4
011: MCO divided by 8
100: MCO divided by 16
others: not allowed

Bits 27:24 **MCOSEL[3:0]**: microcontroller clock output
Set and cleared by software.
0000: MCO output disabled, no clock on MCO
0001: sysclkpre system clock after PLL1RCLKPRE division selected
0011: HSI16 clock selected
0100: HSE32 clock selected
0101: pll1rclk clock selected
0110: LSI clock selected
0111: LSE clock selected
1000: pll1pclk clock selected
1001: pll1qclk clock selected
1010: hclk5 clock selected
others: reserved
Note: This clock output may have some truncated cycles at startup or during MCO clock source switching.

Bits 23:4 Reserved, must be kept at reset value.

Bits 3:2 **SWS[1:0]**: system clock switch status
Set and cleared by hardware to indicate which clock source is used as system clock.
00: HSI16 oscillator used as system clock
01: reserved
10: HSE32 or HSE32/2, as defined by HSEPRE, used as system clock
11: pll1rclk used as system clock

Bits 1:0 **SW[1:0]**: system clock switch
Set and cleared by software to select system clock source (SYSCLK).
Cleared by hardware when entering Stop and Standby modes
When selecting HSE32 directly or indirectly as system clock and HSE32 oscillator clock security fails, cleared by hardware.
00: HSI16 selected as system clock
01: reserved
10: HSE32 or HSE32/2, as defined by HSEPRE, selected as system clock
11: pll1rclk selected as system clock

12.8.4 RCC clock configuration register 2 (RCC_CFGR2)

Address offset: 0x020

Reset value: 0x0000 0000

Access: word, half-word, and byte access

From 0 to 15 wait states are inserted if access occurs when the APB or AHB prescalers values update is on going.

Access to this register can be protected by RCC PRESCSEC and RCC SPRIV or RCC NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	PPRE2[2:0]			Res.	PPRE1[2:0]			Res.	HPRE[2:0]		
					rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **PPRE2[2:0]**: APB2 prescaler
Set and cleared by software to control the division factor of the APB2 clock (pclk2).
0xx: pclk2 = hclk1 not divided
100: pclk2 = hclk1 divided by 2
101: pclk2 = hclk1 divided by 4
110: pclk2 = hclk1 divided by 8
111: pclk2 = hclk1 divided by 16

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **PPRE1[2:0]:** APB1 prescaler

Set and cleared by software to control the division factor of the APB1 clock (pclk1).

0xx: pclk1 = hclk1 not divided

100: pclk1 = hclk1 divided by 2

101: pclk1 = hclk1 divided by 4

110: pclk1 = hclk1 divided by 8

111: pclk1 = hclk1 divided by 16

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **HPRE[2:0]**: AHB1, AHB2, and AHB4 prescaler

Set and cleared by software to control the division factor of the AHB1, AHB2, and AHB4 clock (hclk1).

Caution: The software must limit the incremental frequency step by setting these bits correctly to ensure that the hclk1 maximum incremental frequency step does not exceed the maximum allowed incremental frequency step (for more details, refer to *Table 109: SYSCLK and bus maximum frequency*). After a write operation to these bits and before decreasing the voltage range, this register must be read to be sure that the new value is taken into account.

0xx: hclk1 = SYSCLK not divided

100: hclk1 = SYSCLK divided by 2

101: hclk1 = SYSCLK divided by 4

110: hclk1 = SYSCLK divided by 8

111: hclk1 = SYSCLK divided by 16

12.8.5 RCC clock configuration register 3 (RCC_CFGR3)

Address offset: 0x024

Reset value: 0x0000 0000

Access: word, half-word, and byte access

From 0 to 15 wait states are inserted if access occurs when the APB or AHB prescalers values update is on going.

Access to this register can be protected by RCC PRESCSEC and RCC SPRIV or RCC NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPRE7[2:0]			Res.	Res.	Res.	Res.
									rw	rw	rw				

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **PPRE7[2:0]**: APB7 prescaler
Set and cleared by software to control the division factor of the APB7 clock (pclk7).
0xx: hclk1 not divided
100: hclk1 divided by 2
101: hclk1 divided by 4
110: hclk1 divided by 8
111: hclk1 divided by 16

Bits 3:0 Reserved, must be kept at reset value.

12.8.6 RCC PLL1 configuration register (RCC_PLL1CFGR)

Address offset: 0x028

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access.

Access to this register can be protected by RCC PLL1SEC and RCC SPRIV or RCC NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL1RCLKPRERDY	PLL1RCLKPRESTEP	PLL1RCLKPRE	Res.	PLL1REN	PLL1QEN	PLL1PEN
									r	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	PLL1M[2:0]			Res.	Res.	Res.	PLL1FRACEN	PLL1RGE[1:0]		PLL1SRC[1:0]	
					rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PLL1RCLKPRERDY**: sysclkpre not divided ready.
Set by hardware after PLL1RCLKPRE has been set from divided to not divide, to indicate that the pll1rclk not divided is available on sysclkpre.
0: sysclkpre divided
1: sysclkpre not divided ready

Bit 21 **PLL1RCLKPRESTEP**: pll1rclk clock for SYSCLK prescaler division step selection
Set and cleared by software to control the division step of the pll1rclk clock for SYSCLK.
0: pll1rclk 2-step division
1: pll1rclk 3-step division

Bit 20 **PLL1RCLKPRE**: pll1rclk clock for SYSCLK prescaler division enable
Set and cleared by software to control the division of the pll1rclk clock for SYSCLK.
0: pll1rclk not divided, sysclkpre = pll1rclk
1: pll1rclk divided, sysclkpre = pll1rclk divided

Bit 19 Reserved, must be kept at reset value.

Bit 18 **PLL1REN**: PLL1 DIVR divider output enable
Set and cleared by software to enable the pll1rclk output of the PLL1.
To save power, the PLL1REN and PLL1R bits must be set to 0 when the pll1rclk is not used.
0: pll1rclk output disabled
1: pll1rclk output enabled

Bit 17 **PLL1QEN**: PLL1 DIVQ divider output enable
Set and reset by software to enable the pll1qclk output of the PLL1.
To save power, the PLL1QEN and PLL1Q bits must be set to 0 when the pll1qclk is not used.
0: pll1qclk output disabled
1: pll1qclk output enabled

Bit 16 **PLL1PEN**: PLL1 DIVP divider output enable
Set and reset by software to enable the pll1pclk output of the PLL1.
To save power, the PLL1PEN and PLL1P bits must be set to 0 when the pll1pclk is not used.
0: pll1pclk output disabled
1: pll1pclk output enabled

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:8 **PLL1M[2:0]**: Prescaler for PLL1
Set and cleared by software to configure the prescaler of the PLL1. The VCO1 input frequency is PLL1 input clock frequency/PLL1M.
This field can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
000: division by 1 (bypass)
001: division by 2
010: division by 3
...
111: division by 8

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **PLL1FRACEN**: PLL1 fractional latch enable
Set and reset by software to latch the content of PLL1FRACN into the ΣΔ modulator.
In order to latch the PLL1FRACN value into the ΣΔ modulator, PLL1FRACEN must be set to 0, then set to 1: the transition 0 to 1 transfers the content of PLL1FRACN into the modulator (see *PLL1* for details).

Bits 3:2 **PLL1RGE[1:0]**: PLL1 input frequency range
Set and reset by software to select the proper reference frequency range used for PLL1.
This field can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
00-01-10: PLL1 input (ref_ck) clock range frequency between 4 and 8 MHz
11: PLL1 input (ref_ck) clock range frequency between 8 and 16 MHz

Bits 1:0 **PLL1SRC[1:0]**: PLL1 entry clock source
Set and cleared by software to select PLL1 clock source. This field can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
Cleared by hardware when entering Stop or Standby modes.
00: no clock sent to PLL1
01: reserved
10: HSI16 clock selected as PLL1 clock entry
11: HSE32 clock after HSEPRE divider selected as PLL1 clock entry

Note: In order to save power, when no PLL1 clock is used, the value of PLL1SRC must be 0.

12.8.7 RCC PLL1 dividers register (RCC_PLL1DIVR)

Address offset: 0x034

Reset value: 0x0101 0280

Access: no wait state; word, half-word, and byte access.

Access to this register can be protected by RCC PLL1SEC and RCC SPRIV or RCC NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PLL1R[6:0]							Res.	PLL1Q[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL1P[6:0]							PLL1N[8:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **PLL1R[6:0]**: PLL1 DIVR division factor

Set and reset by software to control the frequency of the pll1rclk clock. Division factors are forbidden if VCO frequency / (2 x (TRUNC(division factor / 2)) > pll1rclk maximum frequency.
These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
0000000: pll1rclk = VCO output frequency
0000001: pll1rclk = VCO output frequency / 2 (default after reset)
0000010: pll1rclk = VCO output frequency / 3
0000011: pll1rclk = VCO output frequency / 4
...
1111111: pll1rclk = VCO output frequency / 128

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **PLL1Q[6:0]**: PLL1 DIVQ division factor

Set and reset by software to control the frequency of the pll1qclk clock. Division factors are forbidden if VCO frequency / (2 x (TRUNC(division factor / 2)) > pll1qclk maximum frequency.
These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
0000000: pll1qclk = VCO output frequency
0000001: pll1qclk = VCO output frequency / 2 (default after reset)
0000010: pll1qclk = VCO output frequency / 3
0000011: pll1qclk = VCO output frequency / 4
...
1111111: pll1qclk = VCO output frequency / 128

Bits 15:9 **PLL1P[6:0]**: PLL1 DIVP division factor

Set and reset by software to control the frequency of the pll1pclk clock. Division factors are forbidden if VCO frequency / (2 x (TRUNC(division factor / 2)) > pll1pclk maximum frequency.
These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
0000000: pll1pclk = VCO output frequency
0000001: pll1pclk = VCO output frequency / 2 (default after reset)
0000010: pll1pclk = VCO output frequency
0000011: pll1pclk = VCO output frequency / 4
...
1111111: pll1pclk = VCO output frequency / 128

Bits 8:0 **PLL1N[8:0]**: Multiplication factor for PLL1 VCO

Set and reset by software to control the multiplication factor of the VCO.

These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).

0x003: multiplication factor for PLL1 VCO = 4

0x004: multiplication factor for PLL1 VCO = 5

0x005: multiplication factor for PLL1 VCO = 6

...

0x080: multiplication factor for PLL1 VCO = 129 (default after reset)

...

0x1FF: multiplication factor for PLL1 VCO = 512

others: reserved

VCO output frequency = F_{ref_ck} x multiplication factor for PLL1 VCO, when fractional value 0 has been loaded into PLL1FRACN, with:

- multiplication factor for PLL1 VCO between 4 and 512
- input frequency F_{ref_ck} between 4 and 16 MHz

12.8.8 RCC PLL1 fractional divider register (RCC_PLL1FRACR)

Address offset: 0x038

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access.

Access to this register can be protected by RCC PLL1SEC and RCC SPRIV or RCC NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL1FRACN[12:0]													Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:3 **PLL1FRACN[12:0]**: Fractional part of the multiplication factor for PLL1 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO.

These bits can be written at any time, allowing dynamic fine-tuning of the PLL1 VCO.

VCO output frequency = F_{ref_ck} x [multiplication factor for PLL1 VCO + (PLL1FRACN / 2^{13})], with:

- Multiplication factor for PLL1 VCO must be between 4 and 512
- PLL1FRACN can be between 0 and 2^{13}- 1
- The input frequency F_{ref_ck} must be between 4 and 16 MHz

To change the used fractional value on the fly even if the PLL1 is enabled, the application must proceed as follows:

- Set the bit PLL1FRACEN to 0
- Write the new fractional value into PLL1FRACN
- Set the bit PLL1FRACEN to 1

Bits 2:0 Reserved, must be kept at reset value.

12.8.9 RCC clock interrupt enable register (RCC_CIER)

Address offset: 0x050

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSI2RDYIE
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL1RDYIE	Res.	HSERDYIE	HSIRDYIE	Res.	LSERDYIE	LSI1RDYIE
									rw		rw	rw		rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LSI2RDYIE**: LSI2 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the LSI2 oscillator stabilization.

Access can be secured by RCC LSISEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

Note: If the LSI2 is ready (LSI2RDY in RCC_CR is 1) before enabling LSI2RDYIE, the LSI2RDYF in RCC_CIFR is not set, and no interrupt is generated."

0: LSI2 ready interrupt disabled

1: LSI2 ready interrupt enabled

Note: This bit is reserved on STM32WBA25/23 devices.

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **PLL1RDYIE**: PLL1 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by PLL1 lock.

Access to the bit can be secured by RCC PLL1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

Note: If the PLL1 is ready (PLL1RDY in RCC_CR is 1) before enabling PLL1RDYIE, the PLL1RDYF in RCC_CIFR is not set, and no interrupt is generated.

0: PLL1 lock interrupt disabled

1: PLL1 lock interrupt enabled

Bit 5 Reserved, must be kept at reset value.

Bit 4 **HSERDYIE**: HSE32 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the HSE32 oscillator stabilization.

Access to the bit can be secured by RCC HSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

Note: If the HSE is ready (HSERDY in RCC_CR is 1) before enabling HSERDYIE, the HSERDYF in RCC_CIFR is not set, and no interrupt is generated.

0: HSE32 ready interrupt disabled

1: HSE32 ready interrupt enabled

Bit 3 **HSIRDYIE**: HSI16 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the HSI16 oscillator stabilization.

Access to the bit can be secured by RCC HSISEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

Note: If the HSI is ready (HSIRDY in RCC_CR is 1) before enabling HSIRDYIE, the HSIRDYF in RCC_CIFR is not set, and no interrupt is generated.

0: HSI16 ready interrupt disabled

1: HSI16 ready interrupt enabled

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LSERDYIE**: LSE ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the LSE oscillator stabilization.

Access to the bit can be secured by RCC LSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

Note: If the LSE is ready (LSERDY in RCC_CR is 1) before enabling LSERDYIE, the LSERDYF in RCC_CIFR is not set, and no interrupt is generated.

0: LSE ready interrupt disabled

1: LSE ready interrupt enabled

Bit 0 **LSI1RDYIE**: LSI1 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the LSI1 oscillator stabilization.

Access to the bit can be secured by RCC LSISEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

Note: If the LSI is ready (LSIRDY in RCC_CR is 1) before enabling LSIRDYIE, the LSIRDYF in RCC_CIFR is not set, and no interrupt is generated.

0: LSI1 ready interrupt disabled

1: LSI1 ready interrupt enabled

12.8.10 RCC clock interrupt flag register (RCC_CIFR)

Address offset: 0x054

Reset value: 0x0000 0000

Access: no wait state; word; half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSI2RDYF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSECSSF	Res.	Res.	Res.	PLL1RDYF	Res.	HSERDYF	HSIRDYF	Res.	LSERDYF	LSI1RDYF
					r				r		r	r		r	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LSI2RDYF**: LSI2 ready interrupt flag

Set by hardware when the LSI2 clock becomes stable and LSI2RDYIE is set.

Cleared by software setting the LSI2RDYC bit.

Access to the bit can be secured by RCC LSISEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: no clock ready interrupt caused by the LSI2 oscillator

1: clock ready interrupt caused by the LSI2 oscillator

Note: This bit is reserved on STM32WBA25/23 devices.

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **HSECSSF**: HSE32 clock security system interrupt flag

Set by hardware when a clock security failure is detected in the HSE32 oscillator.

Cleared by software setting the HSECSSC bit.

Access to the bit can be secured by RCC HSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: no clock security interrupt caused by HSE32 clock failure

1: clock security interrupt caused by HSE32 clock failure

Bits 9:7 Reserved, must be kept at reset value.

Bit 6 **PLL1RDYF**: PLL1 ready interrupt flag

Set by hardware when the PLL1 locks and PLL1RDYIE is set.

Cleared by software setting the PLL1RDYC bit.

Access to the bit can be secured by RCC PLL1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: no clock ready interrupt caused by PLL1 lock

1: clock ready interrupt caused by PLL1 lock

Bit 5 Reserved, must be kept at reset value.

Bit 4 **HSERDYF**: HSE32 ready interrupt flag

Set by hardware when the HSE32 clock becomes stable and HSERDYIE is set.

Cleared by software setting the HSERDYC bit.

Access to the bit can be secured by RCC HSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: no clock ready interrupt caused by the HSE32 oscillator

1: clock ready interrupt caused by the HSE32 oscillator

Bit 3 **HSIRDYF**: HSI16 ready interrupt flag

Set by hardware when the HSI16 clock becomes stable and HSIRDYIE is set in a response to setting HSION (see RCC_CR). When HSION is not set but the HSI16 oscillator is enabled by the peripheral through a clock request, this bit is not set and no interrupt is generated.

Access to the bit can be secured by RCC HSISEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

Cleared by software setting the HSIRDYC bit.

0: no clock ready interrupt caused by the HSI16 oscillator

1: clock ready interrupt caused by the HSI16 oscillator

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LSERDYF**: LSE ready interrupt flag

Set by hardware when the LSE clock becomes stable and LSERDYIE is set.

Cleared by software setting the LSERDYC bit.

Access to the bit can be secured by RCC LSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: no clock ready interrupt caused by the LSE oscillator

1: clock ready interrupt caused by the LSE oscillator

Bit 0 **LSI1RDYF**: LSI1 ready interrupt flag

Set by hardware when the LSI1 clock becomes stable and LSI1RDYIE is set.

Cleared by software setting the LSI1RDYC bit.

Access to the bit can be secured by RCC LSISEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: no clock ready interrupt caused by the LSI1 oscillator

1: clock ready interrupt caused by the LSI1 oscillator

12.8.11 RCC clock interrupt clear register (RCC_CICR)

Address offset: 0x058

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSI2RDYC
															w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSECSSC	Res.	Res.	Res.	PLL1RDYC	Res.	HSERDYC	HSIRDYC	Res.	LSERDYC	LSI1RDYC
					w				w		w	w		w	w

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LSI2RDYC**: LSI2 ready interrupt clear

Writing this bit to 1 clears the LSI2RDYF flag. Writing 0 has no effect.

Access to the bit can be secured by RCC LSISEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

Note: This bit is reserved on STM32WBA25/23 devices.

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **HSECSSC**: High speed external clock security system interrupt clear

Setting this bit clears the HSECSSF flag. Clearing it has no effect.

Access to the bit can be secured by RCC HSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

Bits 9:7 Reserved, must be kept at reset value.

Bit 6 **PLL1RDYC**: PLL1 ready interrupt clear

Setting this bit clears the PLL1RDYF flag. Clearing it has no effect.

Access to the bit can be secured by RCC PLL1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **HSERDYC**: HSE32 ready interrupt clear

Setting this bit clears the HSERDYF flag. Clearing it has no effect.

Access to the bit can be secured by RCC HSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

Bit 3 **HSIRDYC**: HSI16 ready interrupt clear

Setting this bit clears the HSIRDYF flag. Clearing it has no effect.

Access to the bit can be secured by RCC HSISEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LSERDYC**: LSE ready interrupt clear

Setting this bit clears the LSERDYF flag. Clearing it has no effect.

Access to the bit can be secured by RCC LSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

Bit 0 **LSI1RDYC**: LSI1 ready interrupt clear

Setting this bit clears the LSI1RDYF flag. Clearing it has no effect.

Access to the bit can be secured by RCC LSISEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

12.8.12 RCC AHB1 peripheral reset register (RCC_AHB1RSTR)

Address offset: 0x060

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPDMA1RST
			rw												rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **CRCRST**: CRC reset

Set and cleared by software.

Access can be secured by GTZC1_TZSC CRCSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: No effect

1: Reset CRC

Bits 11:1 Reserved, must be kept at reset value.

Bit 0 **LPDMA1RST**: LPDMA1 reset
Set and cleared by software.
Access can be secured by LPDMA1 SECx. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: No effect
1: Reset LPDMA1

12.8.13 RCC AHB2 peripheral reset register (RCC_AHB2RSTR)

Address offset: 0x064

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	XSPI1RST	Res.	Res.	Res.	Res.	Res.	OTFDEC1RST	Res.	PKARST	Res.	Res.	RNGRST	HASHRST	AESRST
		rw						rw		rw			rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOHRST	Res.	Res.	Res.	Res.	GPIOCRST	GPIOBRST	GPIOARST
								rw					rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **XSPI1RST**: XSPI1 and DLYBXS1 reset
Set and cleared by software.
Access can be secured by GTZC1_TZSC XSPI1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: No effect
1: Reset XSPI1 and DLYBXS1
Note: This bit is reserved on STM32WBA23xx devices

Bits 28:24 Reserved, must be kept at reset value.

Bit 23 **OTFDEC1RST**: OTFDEC1 reset
Set and cleared by software.
When TrustZone® is enabled (TZEN = 1) this bit is secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: No effect
1: Reset OTFDEC1
Note: This bit is reserved on STM32WBA23xx devices

Bit 22 Reserved, must be kept at reset value.

Bit 21 **PKARST**: PKA reset
Set and cleared by software.
Access can be secured by GTZC1_TZSC PKASEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: No effect
1: Reset PKA

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **RNGRST**: Random number generator reset
Set and cleared by software.
Access can be secured by GTZC1_TZSC RNGSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: No effect
1: Reset RNG

Bit 17 **HASHRST:** Hash reset
Set and cleared by software.
Access can be secured by GTZC1_TZSC HASHSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: No effect
1: Reset HASH

Bit 16 **AESRST**: AES hardware accelerator reset
Set and cleared by software.
Access can be secured by GTZC1_TZSC AESSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: No effect
1: Reset AES

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHRST**: IO port H reset
Set and cleared by software.
Access can be secured by GPIOH SECx. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: No effect
1: Reset IO port H

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **GPIOCRST**: IO port C reset
Set and cleared by software.
Access can be secured by GPIOC SECx. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: No effect
1: Reset IO port C

Bit 1 **GPIOBRST**: IO port B reset

Set and cleared by software.

Access can be secured by GPIOB SECx. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: No effect

1: Reset IO port B

Bit 0 **GPIOARST**: IO port A reset

Set and cleared by software.

Access can be secured by GPIOA SECx. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: No effect

1: Reset IO port A

12.8.14 RCC AHB4 peripheral reset register (RCC_AHB4RSTR)

Address offset: 0x06C

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4RST	Res.	Res.	Res.	Res.	Res.
										rw					

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **ADC4RST**: ADC4 reset

Set and cleared by software.

Access can be secured by GTZC1_TZSC ADC4SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: No effect

1: Reset ADC4 interface

Bits 4:0 Reserved, must be kept at reset value.

12.8.15 RCC AHB5 peripheral reset register (RCC_AHB5RSTR)

Address offset: 0x070

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PTACONVRST**: PTACONV reset

Set and cleared by software.

Access can be secured by GTZC1_TZSC PTACONVSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: No effect

1: Reset PTACONV

Bit 0 **RADIORST**: 2.4 GHz RADIO reset

Set and cleared by software.

Access can be secured by GTZC1_TZSC RADIOSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: No effect

1: Reset 2.4 GHz RADIO

12.8.16 RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1)

Address offset: 0x074

Reset value: 0x0000 0000

Access: no wait state; word; half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C1RST	Res.	Res.	Res.	Res.	Res.
										rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2RST
															rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **I2C1RST**: I2C1 reset
Set and cleared by software.
Access can be secured by GTZC1_TZSC I2C1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: No effect
1: Reset I2C1

Bits 20:1 Reserved, must be kept at reset value.

Bit 0 **TIM2RST**: TIM2 reset
Set and cleared by software.
Access can be secured by GTZC1_TZSC TIM2SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: No effect
1: Reset TIM2

12.8.17 RCC APB1 peripheral reset register 2 (RCC_APB1RSTR2)

Address offset: 0x078

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2RST	Res.	Res.	Res.	Res.	Res.
										rw					

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2RST**: LPTIM2 reset
Set and cleared by software.
Access can be secured by GTZC1_TZSC LPTIM2SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: No effect
1: Reset LPTIM2

Bits 4:0 Reserved, must be kept at reset value.

12.8.18 RCC APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x07C

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBRST	Res.	Res.	SAI1RST	Res.	Res.	TIM17RST	TIM16RST	Res.
							rw			rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw														

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USBRST**: USB reset

Set and cleared by software.

Access can be secured by GTZC1_TZSC USBSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: No effect

1: Reset USB

Note: This bit is reserved on STM32WBA23xx devices

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **SAI1RST**: SAI1 reset

Set and cleared by software.

Access can be secured by GTZC1_TZSC SAI1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: No effect

1: Reset SAI1

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TIM17RST**: TIM17 reset

Set and cleared by software.

Access can be secured by GTZC1_TZSC TIM17SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: No effect

1: Reset TIM17

Bit 17 **TIM16RST**: TIM16 reset

Set and cleared by software.

Access can be secured by GTZC1_TZSC TIM16SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: No effect

1: Reset TIM16

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1RST**: USART1 reset

Set and cleared by software.

Access can be secured by GTZC1_TZSC USART1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: No effect

1: Reset USART1

Bits 13:0 Reserved, must be kept at reset value.

12.8.19 RCC APB7 peripheral reset register (RCC_APB7RSTR)

Address offset: 0x080

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LPTIM1RST	Res.	Res.	Res.	I2C3RST	LPUART1RST	SPI3RST	Res.	Res.	Res.	SYSCFGRST	Res.
				rw				rw	rw	rw				rw	

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **LPTIM1RST**: LPTIM1 reset

Set and cleared by software.

Access can be secured by GTZC1_TZSC LPTIM1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: No effect

1: Reset LPTIM1

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **I2C3RST**: I2C3 reset

Set and cleared by software.

Access can be secured by GTZC1_TZSC I2C3 SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: No effect

1: Reset I2C3

Bit 6 **LPUART1RST**: LPUART1 reset

Set and cleared by software.

Access can be secured by GTZC1_TZSC LPUART1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: No effect

1: Reset LPUART1

Bit 5 **SPI3RST**: SPI3 reset

Set and cleared by software.

Access can be secured by GTZC1_TZSC SPI3SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: No effect

1: Reset SPI3

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **SYSCFGRST**: SYSCFG reset

Set and cleared by software.

Access can be secured by SYSCFG SYSCFGSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: No effect

1: Reset SYSCFG

Bit 0 Reserved, must be kept at reset value.

12.8.20 RCC AHB1 peripheral clock enable register (RCC_AHB1ENR)

Address offset: 0x088

Reset value: 0x8000 0100

Access: no wait state; word, half-word, and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM1EN	Res.	Res.	Res.	Res.	Res.	Res.	GTZC1EN	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGEN	Res.
rw							rw							rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCEN	Res.	Res.	Res.	FLASHEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPDMA1EN
			rw				rw								rw

Bit 31 **SRAM1EN**: SRAM1 bus clock enable

Set and reset by software.

Access can be secured by GTZC1_MPCBB1 SECx, INVSECSTATE. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: SRAM1 bus clock disabled

1: SRAM1 bus clock enabled

Bits 30:25 Reserved, must be kept at reset value.

Bit 24 **GTZC1EN**: GTZC1 bus clock enable

Set and reset by software.

Can only be accessed secure when device is secure (TZEN = 1). When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: GTZC1 bus clock disabled

1: GTZC1 bus clock enabled

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **RAMCFGEN**: RAMCFG bus clock enable

Set and cleared by software.

Access can be secured by GTZC1_TZSC RAMCFGSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: RAMCFG bus clock disabled

1: RAMCFG bus clock enabled

Bits 16:13 Reserved, must be kept at reset value.

Bit 12 **CRCEN**: CRC bus clock enable

Set and cleared by software.

Access can be secured by GTZC1_TZSC CRCSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: CRC bus clock disabled

1: CRC bus clock enabled

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLASHEN**: FLASH bus clock enable

Set and cleared by software. This bit can be disabled only when the Flash memory is in power down mode.

Can only be accessed secured when the Flash security state is secure. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: FLASH bus clock disabled

1: FLASH bus clock enabled

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **LPDMA1EN**: LPDMA1 bus clock enable

Set and cleared by software.

Access can be secured by LPDMA1 SECx. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LPDMA1 bus clock disabled

1: LPDMA1 bus clock enabled

12.8.21 RCC AHB2 peripheral clock enable register (RCC_AHB2ENR)

Address offset: 0x08C

Reset value: 0x4000 0000

Access: no wait state; word, half-word, and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SRAM2EN	XSPI1EN	Res.	Res.	Res.	Res.	Res.	OTFDEC1EN	Res.	PKAEN	Res.	Res.	RNGEN	HASHEN	AESEN
	rw	rw						rw		rw			rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOHEN	Res.	Res.	Res.	Res.	GPIOCEN	GPIOBEN	GPIOAEN
								rw					rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SRAM2EN:** SRAM2 bus clock enable

Set and cleared by software.

Access can be secured by GTZC1_MPCBB2 SECx, INVSECSTATE. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: SRAM2 bus clock disabled

1: SRAM2 bus clock enabled

Bit 29 **XSPI1EN:** XSPI1 and DLYBXS1 bus clock enable

Set and cleared by software.

Access can be secured by GTZC1_TZSC XSPI1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: XSPI1 and DLYBXS1 bus clock disabled

1: XSPI1 and DLYBXS1 bus clock enabled

Note: This bit is reserved on STM32WBA23xx devices

Bits 28:24 Reserved, must be kept at reset value.

Bit 23 **OTFDEC1EN:** OTFDEC1 bus clock enable

Set and cleared by software.

When TrustZone® is enabled (TZEN = 1) this bit is secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: OTFDEC1 bus clock disabled

1: OTFDEC1 bus clock enabled

Note: This bit is reserved on STM32WBA23xx devices

Bit 22 Reserved, must be kept at reset value.

Bit 21 **PKAEN:** PKA bus clock enable

Set and cleared by software.

Access can be secured by GTZC1_TZSC PKASEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: PKA bus clock disabled

1: PKA bus clock enabled

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **RNGEN:** RNG bus and kernel clocks enable

Set and cleared by software.

Access can be secured by GTZC1_TZSC RNGSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: RNG bus and kernel clocks disabled

1: RNG bus and kernel clocks enabled

Bit 17 **HASHEN**: HASH bus clock enable

Set and cleared by software.
Access can be secured by GTZC1_TZSC HASHSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: HASH bus clock disabled
1: HASH bus clock enabled

Bit 16 **AESEN:** AES bus clock enable

Set and cleared by software.
Access can be secured by GTZC1_TZSC AESSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: AES bus clock disabled
1: AES bus clock enabled

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHEN:** IO port H bus clock enable

Set and cleared by software.
Access can be secured by GPIOH SECx. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: IO port H bus clock disabled
1: IO port H bus clock enabled

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **GPIOCEN:** IO port C bus clock enable

Set and cleared by software.
Access can be secured by GPIOC SECx. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: IO port C bus clock disabled
1: IO port C bus clock enabled

Bit 1 **GPIOBEN:** IO port B bus clock enable

Set and cleared by software.
Access can be secured by GPIOB SECx. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: IO port B bus clock disabled
1: IO port B bus clock enabled

Bit 0 **GPIOAEN:** IO port A bus clock enable

Set and cleared by software.
Access can be secured by GPIOA SECx. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: IO port A bus clock disabled
1: IO port A bus clock enabled

12.8.22 RCC AHB4 peripheral clock enable register (RCC_AHB4ENR)

Address offset: 0x094

Reset value: 0x0000 0000

Access: no wait state; word, half-word half-word, and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4EN	Res.	Res.	PWREN	Res.	Res.
										rw			rw		

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **ADC4EN**: ADC4 bus and kernel clocks enable

Set and cleared by software.

Access can be secured by GTZC1_TZSC ADC4SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: ADC4 bus and kernel clocks disabled

1: ADC4 bus and kernel clocks enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **PWREN**: PWR bus clock enable

Set and cleared by software.

Can only be accessed secure when one or more features in the PWR is/are secure. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: PWR bus clock disabled

1: PWR bus clock enabled

Bits 1:0 Reserved, must be kept at reset value.

12.8.23 RCC AHB5 peripheral clock enable register (RCC_AHB5ENR)

Address offset: 0x098

Reset value: 0x0000 0000

Access: no wait state; word, half-word half-word, and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PTACONVEN**: PTACONV bus clock enable

Set and cleared by software.

Access can be secured by GTZC1_TZSC PTACONVSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: PTACONV bus clock disabled

1: PTACONV bus clock enabled

Bit 0 **RADIOEN**: 2.4 GHz RADIO bus clock enable

Set and cleared by software.

Access can be secured by GTZC1_TZSC RADIOSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: 2.4 GHz RADIO bus clock disabled (The 2.4 GHz RADIO bus clock may still be enabled by STRADIOCLKON)

1: 2.4 GHz RADIO bus clock enabled

Note: Before accessing the 2.4 GHz RADIO sleep timers registers the RADIOCLKRDY bit must be checked.
When RADIOSMEN and STRADIOCLKON are both cleared, the RADIOCLKRDY bit must be re-checked when exiting low-power modes (Sleep and Stop).

12.8.24 RCC APB1 peripheral clock enable register 1 (RCC_APB1ENR1)

Address offset: 0x09C

Reset value: 0x0000 0000

Access: no wait state; word, half-word half-word, and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C1EN	Res.	Res.	Res.	Res.	Res.
										rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2EN
															rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **I2C1EN**: I2C1 bus and kernel clocks enable

Set and cleared by software.

Access can be secured by GTZC1_TZSC I2C1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: I2C1 bus and kernel clocks disabled

1: I2C1 bus and kernel clocks enabled

Bits 20:1 Reserved, must be kept at reset value.

Bit 0 **TIM2EN**: TIM2 bus and kernel clocks enable

Set and cleared by software.

Access can be secured by GTZC1_TZSC TIM2SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: TIM2 bus and kernel clocks disabled

1: TIM2 bus and kernel clocks enabled

12.8.25 RCC APB1 peripheral clock enable register 2 (RCC_APB1ENR2)

Address offset: 0x0A0

Reset value: 0x0000 0000

Access: no wait state; word, half-word half-word, and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2EN	Res.	Res.	Res.	Res.	Res.
										rw					

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2EN:** LPTIM2 bus and kernel clocks enable
Set and cleared by software.
Access can be secured by GTZC1_TZSC LPTIM2SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: LPTIM2 bus and kernel clocks disabled
1: LPTIM2 bus and kernel clocks enabled

Bits 4:0 Reserved, must be kept at reset value.

12.8.26 RCC APB2 peripheral clock enable register (RCC_APB2ENR)

Address offset: 0x0A4

Reset value: 0x0000 0000

Access: word, half-word half-word, and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBEN	Res.	Res.	SAI1EN	Res.	Res.	TIM17EN	TIM16EN	Res.
							rw			rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw														

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USBEN**: USB bus and kernel clocks enable
Set and cleared by software.
Access can be secured by GTZC1_TZSC USBSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: USB bus and kernel clocks disabled
1: USB bus and kernel clocks enabled
Note: This bit is reserved on STM32WBA23xx devices

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **SAI1EN**: SAI1 bus and kernel clocks enable
Set and cleared by software.
Access can be secured by GTZC1_TZSC SAI1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: SAI1 bus and kernel clocks disabled
1: SAI1 bus and kernel clocks enabled

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TIM17EN**: TIM17 bus clock enable
Set and cleared by software.
Access can be secured by GTZC1_TZSC TIM17SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: TIM17 bus clock disabled
1: TIM17 bus clock enabled

Bit 17 **TIM16EN**: TIM16 bus clock enable
Set and cleared by software.
Access can be secured by GTZC1_TZSC TIM16SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: TIM16 bus clock disabled
1: TIM16 bus clock enabled

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1EN**: USART1bus and kernel clocks enable
Set and cleared by software.
Access can be secured by GTZC1_TZSC USART1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: USART1 bus and kernel clocks disabled
1: USART1 bus and kernel clocks enabled

Bits 13:0 Reserved, must be kept at reset value.

12.8.27 RCC APB7 peripheral clock enable register (RCC_APB7ENR)

Address offset: 0x0A8

Reset value: 0x0000 0000

Access: no wait state; word, half-word half-word, and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBEN	Res.	Res.	Res.	Res.	Res.
										rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LPTIM1EN	Res.	Res.	Res.	I2C3EN	LPUART1EN	SPI3EN	Res.	Res.	Res.	SYSCFGEN	Res.
				rw				rw	rw	rw				rw	

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **RTCAPBEN**: RTC and TAMP bus clock enable

Set and cleared by software.

Can only be accessed secure when one or more features in the RTC or TAMP is/are secure. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: RTC bus clock disabled

1: RTC bus clock enabled

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **LPTIM1EN**: LPTIM1 bus and kernel clocks enable

Set and cleared by software.

Access can be secured by GTZC1_TZSC LPTIM1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LPTIM1 bus and kernel clocks disabled

1: LPTIM1 bus and kernel clocks enabled

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **I2C3EN**: I2C3 bus and kernel clocks enable

Set and cleared by software.

Access can be secured by GTZC1_TZSC I2C3SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: I2C3 bus and kernel clocks disabled

1: I2C3 bus and kernel clocks enabled

Bit 6 **LPUART1EN**: LPUART1 bus and kernel clocks enable
Set and cleared by software.
Access can be secured by GTZC1_TZSC LPUART1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: LPUART1 bus and kernel clocks disabled
1: LPUART1 bus and kernel clocks enabled

Bit 5 **SPI3EN**: SPI3 bus and kernel clocks enable
Set and cleared by software.
Access can be secured by GTZC1_TZSC SPI3SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: SPI3 bus and kernel clocks disabled
1: SPI3 bus and kernel clocks enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **SYSCFGEN**: SYSCFG bus clock enable
Set and cleared by software.
Access can be secured by SYSCFG SYSCFGSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: SYSCFG bus clock disabled
1: SYSCFG bus clock enabled

Bit 0 Reserved, must be kept at reset value.

12.8.28 RCC AHB1 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB1SMENR)

Address offset: 0x0B0

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word half-word, and byte access

This register configures the clock gating only when the corresponding peripheral enable bit of the RCC_AHB1ENR register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM1SMEN	Res.	ICACHESMEN	Res.	Res.	Res.	Res.	GTZC1SMEN	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGSMEN	Res.
rw		rw					rw							rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCSMEN	Res.	Res.	Res.	FLASHSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPDMA1SMEN
			rw				rw								rw

Bit 31 **SRAM1SMEN**: SRAM1 bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC1_MPCBB1 SECx, INVSECSTATE. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: SRAM1 bus clock disabled by the clock gating during Sleep and Stop modes
1: SRAM1 bus clock enabled by the clock gating during Sleep and Stop modes

Bit 30 Reserved, must be kept at reset value.

Bit 29 **ICACHESMEN**: ICACHE bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC1_TZSC ICACHE_REGSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: ICACHE bus clock disabled by the clock gating during Sleep and Stop modes
1: ICACHE bus clock enabled by the clock gating during Sleep and Stop modes

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **GTZC1SMEN**: GTZC1 bus clock enable during Sleep and Stop modes
Set and cleared by software.
Can only be accessed secure when one device is secure (TZEN = 1). When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: GTZC1 bus clock disabled by the clock gating during Sleep and Stop modes
1: GTZC1 bus clock enabled by the clock gating during Sleep and Stop modes

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **RAMCFGSMEN**: RAMCFG bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC1_TZSC RAMCFGSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: RAMCFG bus clock disabled by the clock gating during Sleep and Stop modes
1: RAMCFG bus clock enabled by the clock gating during Sleep and Stop modes

Bits 16:13 Reserved, must be kept at reset value.

Bit 12 **CRCSMEN**: CRC bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC1_TZSC CRCSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: CRC bus clock disabled by the clock gating during Sleep and Stop modes
1: CRC bus clock enabled by the clock gating during Sleep and Stop modes

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLASHSMEN**: FLASH bus clock enable during Sleep and Stop modes
Set and cleared by software.
Can only be accessed secured when the Flash security state is secure. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: FLASH bus clock disabled by the clock gating during Sleep and Stop modes
1: FLASH bus clock enabled by the clock gating during Sleep and Stop modes

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **LPDMA1SMEN**: LPDMA1 bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by LPDMA1 SECx. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: LPDMA1 bus clock disabled by the clock gating during Sleep and Stop modes
1: LPDMA1 bus clock enabled by the clock gating during Sleep and Stop modes
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

12.8.29 RCC AHB2 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB2SMENR)

Address offset: 0x0B4

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word half-word, and byte access

This register configures the clock gating only when the corresponding peripheral enable bit of the RCC_AHB2ENR is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SRAM2SMEN	XSPI1SMEN	Res.	Res.	Res.	Res.	Res.	OTFDEC1SMEN	Res.	PKASMEN	Res.	Res.	RNGSMEN	HASHSMEN	AESSMEN
	rw	rw						rw		rw			rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOHSMEN	Res.	Res.	Res.	Res.	GPIOCSMEN	GPIOBSMEN	GPIOASMEN
								rw					rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SRAM2SMEN**: SRAM2 bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC1_MPCBB2 SECx, INVSECSTATE. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: SRAM2 bus clock disabled by the clock gating during Sleep and Stop modes
1: SRAM2 bus clock enabled by the clock gating during Sleep and Stop modes

Bit 29 **XSPI1SMEN**: XSPI1 and DLYBXS1 bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC1_TZSC XSPI1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: XSPI1 and DLYBXS1 bus clock disabled by the clock gating during Sleep and Stop modes
1: XSPI1 and DLYBXS1 bus clock enabled by the clock gating during Sleep and Stop modes
Note: This bit is reserved on STM32WBA23xx devices

Bits 28:24 Reserved, must be kept at reset value.

Bit 23 **OTFDEC1SMEN**: OTFDEC1 bus clock enable during Sleep and Stop modes
Set and cleared by software.
When TrustZone® is enabled (TZEN = 1) this bit is secure and nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: OTFDEC1 bus clock disabled by the clock gating during Sleep and Stop modes
1: OTFDEC1 bus clock enabled by the clock gating during Sleep and Stop modes
Note: This bit is reserved on STM32WBA23xx devices

Bit 22 Reserved, must be kept at reset value.

Bit 21 **PKASMEN**: PKA bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC1_TZSC PKASEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: PKA bus clock disabled by the clock gating during Sleep and Stop modes
1: PKA bus clock enabled by the clock gating during Sleep and Stop modes

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **RNGSMEN:** Random number generator (RNG) bus and kernel clocks enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC1_TZSC RNGSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: RNG bus and kernel clocks disabled by the clock gating during Sleep and Stop modes
1: RNG bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

Bit 17 **HASHSMEN**: HASH bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC1_TZSC HASHSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: HASH bus clock disabled by the clock gating during Sleep and Stop modes
1: HASH bus clock enabled by the clock gating during Sleep and Stop modes

Bit 16 **AESSMEN**: AES bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC1_TZSC AESSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: AES bus clock disabled by the clock gating during Sleep and Stop modes
1: AES bus clock enabled by the clock gating during Sleep and Stop modes

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHSMEN:** IO port H bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GPIOH SECx. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: IO port H bus clock disabled by the clock gating during Sleep and Stop modes
1: IO port H bus clock enabled by the clock gating during Sleep and Stop modes

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **GPIOCSMEN:** IO port C bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GPIOC SECx. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: IO port C bus clock disabled by the clock gating during Sleep and Stop modes
1: IO port C bus clock enabled by the clock gating during Sleep and Stop modes

Bit 1 **GPIOBSMEN:** IO port B bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GPIOB SECx. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: IO port B bus clock disabled by the clock gating during Sleep and Stop modes
1: IO port B bus clock enabled by the clock gating during Sleep and Stop modes

Bit 0 **GPIOASMEN:** IO port A bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GPIOA SECx. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: IO port A bus clock disabled by the clock gating during Sleep and Stop modes
1: IO port A bus clock enabled by the clock gating during Sleep and Stop modes

12.8.30 RCC AHB4 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB4SMENR)

Address offset: 0x0BC

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word half-word, and byte access

This register configures the clock gating only when the corresponding peripheral enable bit of the RCC_AHB4ENR register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4SMEN	Res.	Res.	PWRSMEN	Res.	Res.
										rw			rw		

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **ADC4SMEN**: ADC4 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC1_TZSC ADC4SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: ADC4 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: ADC4 bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **PWRSMEN**: PWR bus clock enable during Sleep and Stop modes

Set and cleared by software.

Can only be accessed secure when one or more features in the PWR is/are secure. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: PWR bus clock disabled by the clock gating during Sleep and Stop modes

1: PWR bus clock enabled by the clock gating during Sleep and Stop modes

Bits 1:0 Reserved, must be kept at reset value.

12.8.31 RCC AHB5 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB5SMENR)

Address offset: 0x0C0

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word half-word, and byte access

This register configures the clock gating only when the corresponding peripheral enable bit of the RCC_AHB5ENR register is set.

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PTACONVSMEN**: PTACONV bus clock enable during Sleep and Stop modes.

Set and cleared by software.

Access can be secured by GTZC1_TZSC PTACONVSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: PTACONV bus clock disabled by the clock gating during Sleep and Stop modes

1: PTACONV bus clock enabled by the clock gating during Sleep and Stop modes

Bit 0 **RADIOSMEN**: 2.4 GHz RADIO bus clock enable during Sleep and Stop modes when the 2.4 GHz RADIO is active.

Set and cleared by software.

Access can be secured by GTZC1_TZSC RADIOSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: 2.4 GHz RADIO bus clock disabled by the clock gating during Sleep and Stop modes (The 2.4 GHz RADIO bus clock may still be enabled by STRADIOCLKON)

1: 2.4 GHz RADIO bus clock enabled by the clock gating during Sleep and Stop modes when the 2.4 GHz RADIO is active

12.8.32 RCC APB1 peripheral clocks enable in Sleep and Stop modes register 1 (RCC_APB1SMENR1)

Address offset: 0x0C4

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word half-word, and byte access

This register configures the clock gating only when the corresponding peripheral enable bit of the RCC_APB1ENR1 register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C1SMEN	Res.	Res.	Res.	Res.	Res.
										rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2SMEN
															rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **I2C1SMEN**: I2C1 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC1_TZSC I2C1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: I2C1 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: I2C1 bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bits 20:1 Reserved, must be kept at reset value.

Bit 0 **TIM2SMEN**: TIM2 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC1_TZSC TIM2SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: TIM2 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: TIM2 bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

12.8.33 RCC APB1 peripheral clocks enable in Sleep and Stop modes register 2 (RCC_APB1SMENR2)

Address offset: 0x0C8

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word half-word, and byte access

This register configures the clock gating only when the corresponding peripheral enable bit of the RCC_APB1ENR2 register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2SMEN	Res.	Res.	Res.	Res.	Res.
										rw					

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2SMEN:** LPTIM2 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC1_TZSC LPTIM2SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LPTIM2 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: LPTIM2 bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bits 4:0 Reserved, must be kept at reset value.

12.8.34 RCC APB2 peripheral clocks enable in Sleep and Stop modes register (RCC_APB2SMENR)

Address offset: 0x0CC

Reset value: 0xFFFF FFFF

Access: word, half-word half-word, and byte access

This register configures the clock gating only when the corresponding peripheral enable bit of the RCC_APB2ENR register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBSMEN	Res.	Res.	SAI1SMEN	Res.	Res.	TIM17SMEN	TIM16SMEN	Res.
							rw			rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw														

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USBSMEN**: USB bus and kernel clocks enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC1_TZSC USBSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: USB bus and kernel clocks disabled by the clock gating during Sleep and Stop modes
1: USB bus and kernel clocks enabled by the clock gating during Sleep and Stop modes
Note: This bit is reserved on STM32WBA23xx devices

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **SAI1SMEN**: SAI1 bus and kernel clocks enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC1_TZSC SAI1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: SAI1 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes
1: SAI1 bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TIM17SMEN**: TIM17 bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC1_TZSC TIM17SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: TIM17 bus clock disabled by the clock gating during Sleep and Stop modes
1: TIM17 bus clock enabled by the clock gating during Sleep and Stop modes

Bit 17 **TIM16SMEN**: TIM16 bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC1_TZSC TIM16SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: TIM16 bus clock disabled by the clock gating during Sleep and Stop modes
1: TIM16 bus clock enabled by the clock gating during Sleep and Stop modes

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1SMEN**: USART1 bus and kernel clocks enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC1_TZSC USART1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0: USART1 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes
1: USART1 bus and kernel clocks enabled by the clock gating during Sleep and Stop modes
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bits 13:0 Reserved, must be kept at reset value.

12.8.35 RCC APB7 peripheral clock enable in Sleep and Stop modes register (RCC_APB7SMENR)

Address offset: 0x0D0

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word half-word, and byte access

This register configures the clock gating only when the corresponding peripheral enable bit of the RCC_APB7ENR register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBSMEN	Res.	Res.	Res.	Res.	Res.
										rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LPTIM1SMEN	Res.	Res.	Res.	I2C3SMEN	LPUART1SMEN	SPI3SMEN	Res.	Res.	Res.	SYSCFGSMEN	Res.
				rw				rw	rw	rw				rw	

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **RTCAPBSMEN**: RTC and TAMP APB clock enable during Sleep and Stop modes

Set and cleared by software.

Can only be accessed secure when one or more features in the RTC or TAMP is/are secure. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: RTC and TAMP APB clock disabled by the clock gating during Sleep and Stop modes

1: RTC and TAMP APB clock enabled by the clock gating during Sleep and Stop modes

Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **LPTIM1SMEN**: LPTIM1 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC1_TZSC LPTIM1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LPTIM1 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: LPTIM1 bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **I2C3SMEN**: I2C3 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software..

Access can be secured by GTZC1_TZSC I2C3SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: I2C3 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: I2C3 bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 6 **LPUART1SMEN**: LPUART1 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC1_TZSC LPUART1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LPUART1 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: LPUART1 bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 5 **SPI3SMEN**: SPI3 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC1_TZSC SPI3SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: SPI3 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: SPI3 bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **SYSCFGSMEN**: SYSCFG bus clock enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by SYSCFG SYSCFGSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: SYSCFG bus clock disabled by the clock gating during Sleep and Stop modes

1: SYSCFG bus clock enabled by the clock gating during Sleep and Stop modes

Bit 0 Reserved, must be kept at reset value.

12.8.36 RCC peripherals independent clock configuration register 1 (RCC_CCIPR1)

Address offset: 0x0E0

Reset value: 0x0000 0000

Access: no wait states; word, half-word half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMICSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSTICKSEL[1:0]		Res.	Res.	LPTIM2SEL[1:0]		Res.	Res.
rw								rw	rw			rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	I2C1SEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART1SEL[1:0]	
				rw	rw									rw	rw

Bit 31 **TIMICSEL**: Clocks sources for TIM16, TIM17, and LPTIM2 internal input capture

When the TIMICSEL bit is set, the TIM16, TIM17, and LPTIM2 internal input capture can be connected to HSI16/256.

When TIMICSEL is cleared, the HSI16, clock sources cannot be selected as TIM16, TIM17, or LPTIM2 internal input capture.

Access can be secured by GTZC_TZSC TIM16SEC or GTZC_TZSC TIM17SEC, LPTIM2SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: HSI16 divider disabled

1: HSI16/256 generated and can be selected by TIM16, TIM17 and LPTIM2 as internal input capture

Note: The clock division must be disabled (TIMICSEL configured to 0) before selecting or changing a clock sources division.

Bits 30:24 Reserved, must be kept at reset value.

Bits 23:22 **SYSTICKSEL[1:0]**: SysTick clock source selection

These bits are used to select the SysTick clock source.

Access can be secured by RCC SYSCLKSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

00: hclk1 divided by 8 selected

01: LSI selected

10: LSE selected

11: HSI16 divided by 4

Note: When HSI16, LSE or LSI is selected, the AHB frequency must be at least four times higher than the LSI or LSE frequency. In addition, a jitter up to one hclk1 cycle is introduced, due to the LSE or LSI sampling with hclk1 in the SysTick circuitry.

Bits 21:20 Reserved, must be kept at reset value.

Bits 19:18 **LPTIM2SEL[1:0]**: Low-power timer 2 kernel clock source selection

These bits are used to select the LPTIM2 kernel clock source.

Access can be secured by GTZC1_TZSC LPTIM2SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

00: pclk1 selected

01: LSI selected

10: HSI16 selected

11: LSE selected

Note: The LPTIM2 is functional in Stop 0 and Stop 1 mode only when the kernel clock is LSI, LSE or HSI16 if HSIKERON = 1.

Bits 17:12 Reserved, must be kept at reset value.

Bits 11:10 **I2C1SEL[1:0]**: I2C1 kernel clock source selection

These bits are used to select the I2C1 kernel clock source.

Access can be secured by GTZC1_TZSC I2C1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

00: pclk1 selected

01: SYSCLK selected

10: HSI16 selected

11: reserved

Note: The I2C1 is functional in Stop 0 and Stop 1 mode only when the kernel clock is HSI16.

Bits 9:2 Reserved, must be kept at reset value.

Bits 1:0 **USART1SEL[1:0]**: USART1 kernel clock source selection

This bits are used to select the USART1 kernel clock source.

Access can be secured by GTZC1_TZSC USART1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

00: pclk2 selected

01: SYSCLK selected

10: HSI16 selected

11: LSE selected

Note: The USART1 is functional in Stop 0 and Stop 1 mode only when the kernel clock is HSI16 or LSE.

12.8.37 RCC peripherals independent clock configuration register 2 (RCC_CCIPR2)

Address offset: 0x0E4

Reset value: 0x0000 0000

Access: no wait state; word, half-word half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ASSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	RNGSEL[1:0]		Res.	Res.	Res.	Res.	SAI1SEL[2:0]			Res.	Res.	Res.	Res.	Res.
		rw	rw					rw	rw	rw					

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ASSEL**: RCC audio synchronization kernel clock source selection

This bit allow the selection of the audio synchronization kernel clock source.

Access can be secured by GTZC_TZSC SAI1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecurenonsecure with RCC NSPRIV.

0: pll1pclk selected

1: pll1qclk selected

Bits 29:14 Reserved, must be kept at reset value.

Bits 13:12 **RNGSEL[1:0]**: RNG kernel clock source selection

These bits allow the selection of the RNG kernel clock source.

Access can be secured by GTZC1_TZSC RNGSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

00: LSE selected

01: LSI selected

10: HSI16 selected

11: pll1rclk divide by 2 selected

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:5 **SAI1SEL[2:0]**: SAI1 kernel clock source selection

These bits allow the selection of select the SAI1 kernel clock source.

Access can be secured by GTZC1_TZSC SAI1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

000: pll1pclk selected

001: pll1qclk selected

010: SYSCLK selected

011: input pin AUDIOCLK selected

100: HSI16 clock selected

others reserved

Note: If the selected clock is the external AUDIOCLK and this clock is stopped, a switch to another source is impossible.

Bits 4:0 Reserved, must be kept at reset value.

12.8.38 RCC peripherals independent clock configuration register 3 (RCC_CCIPR3)

Address offset: 0x0E8

Reset value: 0x0000 0000

Access: no wait state; word, half-word half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ADCSEL[2:0]			LPTIM1SEL[1:0]		Res.	Res.	I2C3SEL[1:0]		Res.	SPI3SEL[1:0]		Res.	LPUART1SEL[1:0]	
	rw	rw	rw	rw	rw			rw	rw		rw	rw		rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:12 **ADCSEL[2:0]**: ADC4 kernel clock source selection

These bits are used to select the ADC4 kernel clock source.

Access can be secured by GTZC1_TZSC ADC4SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

000: hclk1 clock selected

001: SYSCLK selected

010: pll1pclk selected

011: HSE32 clock selected

100: HSI16 clock selected

others: reserved

Note: The ADC4 is functional in Stop 0 and Stop 1 modes only when the kernel clock is HSI16.

Bits 11:10 **LPTIM1SEL[1:0]**: LPTIM1 kernel clock source selection

These bits are used to select the LPTIM1 kernel clock source.

Access can be secured by GTZC1_TZSC LPTIM1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

00: pclk7 selected.

01: LSI selected

10: HSI16 selected

11: LSE selected

Note: The LPTIM1 is functional in Stop 0, 1, and 2 modes only when the kernel clock is LSI, LSE, HSI16 with HSIKERON = 1.

Bits 9:8 Reserved, must be kept at reset value.

Bits 7:6 **I2C3SEL[1:0]**: I2C3 kernel clock source selection

These bits are used to select the I2C3 kernel clock source.

Access can be secured by GTZC1_TZSC I2C3SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

00: pclk7 selected

01: SYSCLK selected

10: HSI16 selected

11: reserved

Note: The I2C3 is functional in Stop 0, 1, and 2 modes only when the kernel clock is HSI16

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **SPI3SEL[1:0]**: SPI3 kernel clock source selection

These bits are used to select the SPI3 kernel clock source.

Access can be secured by GTZC1_TZSC SPI3SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

00: pclk7 selected

01: SYSCLK selected

10: HSI16 selected

11: reserved

Note: The SPI3 is functional in Stop 0, 1, and 2 mode only when the kernel clock is HSI16.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **LPUART1SEL[1:0]**: LPUART1 kernel clock source selection

These bits are used to select the LPUART1 kernel clock source.

Access can be secured by GTZC1_TZSC LPUART1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

00: pclk7 selected

01: SYSCLK selected

10: HSI16 selected

11: LSE selected

Note: The LPUART1 is functional in Stop 0, 1, and 2 modes only when the kernel clock is HSI16 or LSE.

12.8.39 RCC backup domain control register (RCC_BDCR1)

Address offset: 0x0F0

Reset value: 0x0000 X008

Reset by a backup domain VDD reset and BDRST.

Where X (LSETRIM) is loaded with a factory-programmed value at BOR0 reset and OBL_LAUNCH when SBF is cleared.

Fields LSETRIM[1:0] and BDRST are reset only by backup domain power-on reset (BOR0), not reset by a BDRST reset.

Reset value is not affected by exiting Standby mode, nor by a system reset or BORx (x = 1 to 4). This does not apply to the LSIGATEEN and LSEGATEEN fields, which are also reset on a system reset.

Access: 0 ≤ wait state ≤ 3; word, half-word half-word, and byte access

Wait states are inserted in case of successive access operations to this register.

Note: The bits of this register, except BDRST, are outside the core domain. As a result, after reset, these bits are write-protected and the DBP bit in the Section 11.10.9: PWR disable backup domain register (PWR_DBPR) must be set before these can be modified (see Section 11: Power control (PWR) for further information).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LSICLK	LSI2RDY	LSI2ON	LSI1PREDIV	LSI1RDY	LSI1ON	LSCOSEL	LSCOEN	LSEGATED	LSIGATED	LSEGATEEN	LSIGATEEN	RADIOSTSEL[1:0]		Res.	BDRST
r	r	rw	rw	r	rw	rw	rw	r	r	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LSETRIM[1:0]		LSEGFON	LSESYSRDY	Res.	RTCSEL[1:0]		LSESYSEN	LSECSSD	LSECSSON	LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
	rw	rw	rw	r		rw	rw	rw	r	rw	rw	rw	rw	r	rw

Bit 31 **LSICLK**: LSI Low-speed clock signal **(non user)**

Access can be secured by RCC LSISEC. When secure, nonsecure read access is RAZ. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LSI clock signal low

1: LSI clock signal high

Bit 30 **LSI2RDY**: LSI2 oscillator ready

Set and cleared by hardware to indicate when the LSI2 oscillator is stable. After the LSI2ON bit is cleared, LSI2RDY goes low after three internal low-speed oscillator clock cycles.

Access can be secured by RCC LSISEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LSI2 oscillator not ready

1: LSI2 oscillator ready

Note: This bit is reserved on STM32WBA25/23 devices.

Bit 29 **LSI2ON**: LSI2 oscillator enable

Set and cleared by software.

Access can be secured by RCC LSISEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LSI2 oscillator off

1: LSI2 oscillator on

Note: This bit is reserved on STM32WBA25/23 devices.

Bit 28 **LSI1PREDIV**: LSI1 Low-speed clock divider configuration

Set and cleared by software to enable the LSI1 division. This bit can be written only when the LSI1 is disabled (LSI1ON = 0 and LSI1RDY = 0). The LSI1PREDIV cannot be changed if the LSI1 is used by the IWDG or by the RTC.

LSI1PREDIV write protection can be disabled by LSIKEYR.

Access can be secured by RCC LSISEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LSI1 not divided

1: LSI1 divided by 128

Bit 27 **LSI1RDY**: LSI1 oscillator ready

Set and cleared by hardware to indicate when the LSI1 oscillator is stable. After the LSI1ON bit is cleared, LSI1RDY goes low after three internal low-speed oscillator clock cycles. This bit is set when the LSI1 is used by IWDG or RTC, even if LSI1ON = 0.

Access can be secured by RCC LSISEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LSI1 oscillator not ready

1: LSI1 oscillator ready

Bit 26 **LSI1ON**: LSI1 oscillator enable

Set and cleared by software.

Access can be secured by RCC LSISEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LSI1 oscillator off

1: LSI1 oscillator on

Bit 25 **LSCOSEL**: Low-speed clock output selection

Set and cleared by software.

Access can be secured by RCC LSISEC and/or RCC LSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LSI clock selected

1: LSE clock selected

Bit 24 **LSCOEN**: Low-speed clock output (LSCO) enable

Set and cleared by software.

Access can be secured by RCC LSISEC and/or RCC LSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LSCO disabled

1: LSCO enabled

Bit 23 **LSEGATED**: Digital domain low-speed LSE clock rated

Set and cleared by software.

LSE clock, when used by a peripheral, shall be gated before entering in Stop 3 mode. Software shall wait for LSEGATED to be set before entering Stop 3 mode.

Access can be secured by RCC LSESEC. When secure, a nonsecure read/write access is RAZ. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: digital domain LSE clock not gated

1: digital domain LSE clock gated off

Note: When RTC select LSE as clock in RTCSEL and LSE clock is gated off, RTC will be reset.

Bit 22 **LSIGATED**: Digital domain low-speed LSI clock gated

Set and cleared by software.

LSI clock, when used by a peripheral, shall be gated before entering in Stop 3 mode. Software shall wait for LSIGATED to be set before entering Stop 3 mode.

Access can be secured by RCC LSISEC. When secure, nonsecure read/write access is RAZ. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: digital domain LSI clock not gated

1: digital domain LSI clock gated off

Note: When RTC select LSI as clock in RTCSEL and LSI clock is gated off, RTC will be reset.

Bit 21 **LSEGATEEN**: Digital domain low-speed LSE clock gating requested

Set and cleared by software.

LSE clock, when used by a peripheral, shall be gated before entering in Stop 3 mode.

Access can be secured by RCC LSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: digital domain no LSE clock gating request

1: digital domain LSE clock gating request

Bit 20 **LSIGATEEN**: Digital domain low-speed LSI clock gating requested

Set and cleared by software.

LSI clock, when used by a peripheral, shall be gated before entering in Stop 3 mode.

Access can be secured by RCC LSISEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: digital domain no LSI clock gating request

1: digital domain LSI clock gating request

Bits 19:18 **RADIOSTSEL[1:0]**: 2.4 GHz RADIO sleep timer kernel clock enable and selection

Set and cleared by software.

Access can be secured by GTZC1_TZSC RADIOSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

00: no clock selected, 2.4 GHz RADIO sleep timer kernel clock disabled

01: LSE oscillator clock selected

10: Reserved

11: HSE32 oscillator clock divided by 1000 selected

Bit 17 Reserved, must be kept at reset value.

Bit 16 **BDRST**: Backup domain software reset

Set and cleared by software.

A backup domain reset is generated only when the backup domain access protection is disabled.

Can only be accessed secure when one or more features in the RTC or TAMP are secure. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: Reset not activated

1: Reset the entire Backup domain

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **LSETRIM[1:0]**: LSE trimming

These bits are initialized at startup and after OBL_LAUNCH with SBF cleared with the factory-programmed LSE calibration value.

Set and cleared by software. These bits must be modified only once after a BOR reset or an OBL_LAUNCH and before enabling LSE with LSEON (when both LSEON = 0 and LSERDY = 0).

Access can be secured by RCC LSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

00: current source resistance 5/4 x R

01: current source resistance R

10: current source resistance 3/4 x R

11: current source resistance 2/3 x R

Note: OBL_LAUNCH of this field occurs only when SBF is cleared and must then only be started by software when LSE oscillator is disabled, LSEON = 0 and LSERDY = 0.

Bit 12 **LSEGFON**: LSE clock glitch filter enable

Set and cleared by hardware to enable the LSE glitch filter. This bit can be written only when the LSE is disabled (LSEON = 0 and LSERDY = 0).

Access can be secured by RCC LSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LSE glitch filter disabled

1: LSE glitch filter enabled

Bit 11 **LSESYSRDY**: LSE system clock (LSESYS) ready

Set and cleared by hardware to indicate when the LSE system clock is stable. When the LSESYSEN bit is set, the LSESYSRDY flag is set after two LSE clock cycles.

The LSE clock must be already enabled and stable (LSEON and LSERDY are set).

When the LSEON bit is cleared, LSERDY goes low after six external low-speed oscillator clock cycles.

Access can be secured by RCC LSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LSESYS clock not ready

1: LSESYS clock ready

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **RTCSEL[1:0]**: RTC and TAMP kernel clock source enable and selection

Set by software to enable and select the clock source for the RTC.

Can only be accessed secure when one or more features in the RTC or TAMP is/are secure. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

00: no clock selected, RTC and TAMP kernel clock disabled

01: LSE oscillator clock selected, and enabled

10: LSI oscillator clock selected, and enabled

11: HSE32 oscillator clock divided by 32 selected, and enabled

Bit 7 **LSESYSEN**: LSE system clock (LSESYS) enable

Set by software to enable the LSE system clock generated by RCC. The lsesys clock is used for peripherals (USART, LPUART, LPTIM, RNG, 2.4 GHz RADIO) and functions (LSCO, MCO, TIM triggers, LPTIM trigger) excluding the RTC, TAMP and LSECSS.

Access can be secured by RCC LSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LSESYS clock disabled

1: LSESYS clock enabled

Bit 6 **LSECSSD**: Low speed external clock security, LSE failure Detection

Set by hardware to indicate when a failure is detected by the LSECCS on the external 32 kHz oscillator.

Reset when LSCSSON bit is cleared.

Access can be secured by RCC LSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: no failure detected on LSE

1: failure detected on LSE

Bit 5 **LSECSSON**: Low speed external clock security enable

Set by software to enable the LSECSS. LSECSSON must be enabled after the LSE oscillator is enabled (LSEON bit enabled) and ready (LSERDY flag set by hardware) and after the RTCSEL bit is selected.

Once enabled, this bit cannot be disabled, except after a LSE failure detection (LSECSSD = 1). In that case, the software must disable the LSECSSON bit.

Access can be secured by RCC LSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LSECSS disabled off

1: LSECSS enabled on

Bits 4:3 **LSEDRV[1:0]**: LSE oscillator drive capability

Set by software to modulate the drive capability of the LSE oscillator. LSEDRV must be programmed to a different value than 0 before enabling the LSE oscillator in ‘Xtal’ mode.

Access can be secured by RCC LSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

00: reserved

01: ‘Xtal mode’ medium-low driving capability

10: ‘Xtal mode’ medium-high driving capability

11: ‘Xtal mode’ higher driving capability

Note: The oscillator is in ‘Xtal mode’ when it is not in bypass mode.

Bit 2 **LSEBYP**: LSE oscillator bypass

Set and cleared by software to bypass oscillator. This bit can be written only when the external 32 kHz oscillator is disabled (LSEON = 0 and LSERDY = 0).

Access can be secured by RCC LSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LSE oscillator 'Xtal' mode

1: LSE oscillator bypassed

Bit 1 **LSERDY**: LSE oscillator ready

Set and cleared by hardware to indicate when the external 32 kHz oscillator is stable. After the LSEON bit is cleared, LSERDY goes low after six external low-speed oscillator clock cycles.

Access can be secured by RCC LSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LSE oscillator not ready

1: LSE oscillator ready

Bit 0 **LSEON**: LSE oscillator enable

Set and cleared by software.

Access can be secured by RCC LSESEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.

0: LSE oscillator off

1: LSE oscillator on

12.8.40 RCC Backup domain control register (RCC_BDCR2)

Address offset: 0x0F8

Reset value: 0x0000 0000

(reset value not effected by exit Standby mode, nor by system reset or BORx (x = 1 to 4).

Access: 0 ≤ wait state ≤ 3; word, half-word, and byte access

Wait states are inserted in case of successive accesses to this register.

Access to this register can be protected by RCC SPRIV or RCC NSPRIV.

The bits of this register are outside the Core domain. As a result, after reset, they are write protected and the DBP bit in the *PWR disable backup domain register (PWR_DBPR)* must be set before they can be modified (see *Section 11: Power control (PWR)* for further information).

Note: This register is reserved on STM32WBA25/23 devices.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSI2CFG[3:0]				Res.	LSI2MODE[2:0]		
								rw	rw	rw	rw		rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **LSI2CFG**: LSI2 oscillator configuration
Set and cleared by software to control the temperature at which the frequency temperature sensitivity is close to 0.
Access can be secured by RCC LSISEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0x0: LSI2 frequency temperature sensitivity is close to 0 at +80 oC
0x1: LSI2 frequency temperature sensitivity is close to 0 at +50 oC
0x2: LSI2 frequency temperature sensitivity is close to 0 at +20 oC
Others: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **LSI2MODE**: LSI2 oscillator operating mode configuration
Set and cleared by software to select operating mode of power consumption versus accuracy.
Access can be secured by RCC LSISEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when nonsecure with RCC NSPRIV.
0b000: nominal-power, high accuracy
0b001: low-power, medium accuracy
0b010: ultra-low-power, low accuracy
Others: Reserved

12.8.41 RCC control/status register (RCC_CSR)

Address offset: 0x0F4

Reset value: 0x0C00 0000

Reset flags are only reset by BOR0 power reset.

Access: 0 ≤ wait state ≤ 3; word, half-word half-word, and byte access

Wait states are inserted in case of successive accesses to this register.

Access to this register can be protected by RCC SPRIV or RCC NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWRRSTF	Res.	IWDGRSTF	SFTRSTF	BORRSTF	PINRSTF	OBLRSTF	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r		r	r	r	r	r		rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 **LPWRRSTF**: Low-power reset flag
Set by hardware when a reset occurs due to illegal Stop and Standby modes entry.
Cleared by writing to the RMVF bit.
0: no illegal mode reset occurred
1: illegal mode reset occurred

Bit 30 Reserved, must be kept at reset value.

Bit 29 **IWDGRSTF**: Independent watchdog reset flag
Set by hardware when an independent watchdog reset domain occurs.
Cleared by writing to the RMVF bit.
0: no independent watchdog reset occurred
1: independent watchdog reset occurred

Bit 28 **SFTRSTF**: Software reset flag
Set by hardware when a software reset occurs.
Cleared by writing to the RMVF bit.
0: no software reset occurred
1: software reset occurred

Bit 27 **BORRSTF**: BOR flag
Set by hardware when a BOR occurs.
Cleared by writing to the RMVF bit.
0: no BOR occurred
1: BOR occurred

Bit 26 **PINRSTF**: NRST pin reset flag
Set by hardware when a reset from the NRST pin occurs.
Cleared by writing to the RMVF bit.
0: No reset from NRST pin occurred
1: Reset from NRST pin occurred

Bit 25 **OBLRSTF**: Option byte loader reset flag
Set by hardware when a reset from the option byte loading occurs.
Cleared by writing to the RMVF bit.
0: No reset from option byte loading occurred
1: Reset from option byte loading occurred

Bit 24 Reserved, must be kept at reset value.

Bit 23 **RMVF**: Remove reset flag
Set by software to clear the reset flags.
Access can be secured by RCC RMVFSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt.
0: No effect
1: Clear the reset flags (once set by software bit is cleared automatically by hardware)

Bits 22:0 Reserved, must be kept at reset value.

12.8.42 RCC secure configuration register (RCC_SECCFGR)

Address offset: 0x110

Reset value: 0x0000 0000

Access: no wait state; word, half-word half-word, and byte access

When the system is secure (TZEN = 1), this register can be written only through secure privileged access if RCC SPRIV = 1 and through secure privileged or unprivileged access if RCC SPRIV = 0. Nonsecure write access generates an illegal access event and the data is

not written. This register can be read by secure or nonsecure, privileged or unprivileged access.

When the system is not secure (TZEN = 0), this register is read as 0 and the register write is ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RMVFSEC	Res.	Res.	Res.	Res.	PLL1SEC	PRESCSEC	SYSCLKSEC	LSESEC	LSISEC	Res.	HSESEC	HSISEC
			rw					rw	rw	rw	rw	rw		rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **RMVFSEC**: Remove reset flag security
Set and reset by software.
0: Nonsecure
1: Secure

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **PLL1SEC**: PLL1 clock configuration and status bits security
Set and reset by software.
0: Nonsecure
1: Secure

Bit 6 **PRESCSEC**: AHBx/APBx prescaler configuration bits security
Set and reset by software.
0: Nonsecure
1: Secure

Bit 5 **SYSCLKSEC**: SYSCLK selection, clock output on MCO configuration security
Set and reset by software.
0: Nonsecure
1: Secure

Bit 4 **LSESEC**: LSE clock configuration and status bits security
Set and reset by software.
0: Nonsecure
1: Secure

Bit 3 **LSISEC**: LSI clock configuration and status bits security
Set and reset by software.
0: Nonsecure
1: Secure

Bit 2 Reserved, must be kept at reset value.

Bit 1 **HSESEC**: HSE32 clock configuration bits, status bits and HSECSS security
Set and reset by software.
0: Nonsecure
1: Secure

Bit 0 **HSISEC**: HSI16 clock configuration and status bits security
Set and reset by software.
0: Nonsecure
1: Secure

12.8.43 RCC privilege configuration register (RCC_PRIVCFGR)

Address offset: 0x114

Reset value: 0x0000 0000

Access: no wait state; word, half-word half-word, and byte access

This register can be written only by a privileged access. It can be read by privileged or unprivileged access.

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **NSPRIV**: RCC nonsecure functions privilege configuration
Set and reset by software.
This bit can be written only by privileged access, secure or nonsecure.
0: Read and write to RCC nonsecure functions can be done by privileged or unprivileged access.
1: Read and write to RCC nonsecure functions can be done by privileged access only.

Bit 0 **SPRIV**: RCC secure functions privilege configuration
Set and reset by software.
This bit can be written only by a secure privileged access.
0: Read and write to RCC secure functions can be done by privileged or unprivileged access.
1: Read and write to RCC secure functions can be done by privileged access only.

12.8.44 RCC audio synchronization control register (RCC_ASCR)

Address offset: 0x1C0

Reset value: 0x0000 4000

Access to this register can be protected by GTZC_TZSC SAI1SEC and RCC SPRIV or RCC NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TPS[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PSC[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEN
	rw	rw	rw	rw	rw	rw	rw								rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **TPS[6:0]:** Capture prescaler
This field is set and cleared by software.
Capture period in number of counter periods. Capture period = counter period * (TPS + 1).

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **PSC[6:0]:** Clock prescaler
This field is set and cleared by software.
This field can only be written by software when the counter is disabled (RCC_ASCR.CEN = 0)
Counter clock frequency = $f_{audiosync_ker_ck}$ / (PSC + 1).

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **CEN:** Counter enable
This bit is set and cleared by software.
Clearing this bit will reset the audio synchronization counter and capture prescaler and all associated registers ASCR, ASIER, ASSR, ASCNTR, ASARR, ASCAR, and ASCOR.
0: Audio synchronization counter and kernel clock disabled
1: Audio synchronization counter and kernel clock enabled

12.8.45 RCC audio synchronization interrupt enable register (RCC_ASIER)

Address offset: 0x1C4

Reset value: 0x0000 0000

Access to this register can be protected by GTZC_TZSC SAI1SEC and RCC SPRIV or RCC NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARIE	CAEIE	COIE	CAIE
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ARIE:** Auto-reload interrupt enable
This bit is set and cleared by software.
0: Auto-reload interrupt disabled and flag masked
1: Auto-reload interrupt and flag enabled

Bit 2 **CAEIE:** Capture error interrupt enable
This bit is set and cleared by software.
0: Capture error interrupt disabled and flag masked
1: Capture error interrupt and flag enabled

Bit 1 **COIE:** Comparer interrupt enable
This bit is set and cleared by software.
0: Compare interrupt disabled and flag masked
1: Compare interrupt and flag enabled

Bit 0 **CAIE:** Capture trigger interrupt enable
This bit is set and cleared by software.
0: Capture trigger interrupt disabled and flag masked
1: Capture trigger interrupt and flag enabled

12.8.46 RCC audio synchronization status register (RCC_ASSR)

Address offset: 0x1C8

Reset value: 0x0000 0000

Access to this register can be protected by GTZC_TZSC SAI1SEC and RCC SPRIV or RCC NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARF	CAEF	COF	CAF
												rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ARF:** Auto-reload interrupt flag
This bit is set by hardware, only when ARIE is enabled. This bit is cleared by software by writing it to ‘0’ or masked when ARIE is ‘0’.
0: No auto-reload has been detected
1: Auto-reload has been done

Bit 2 **CAEF:** Capture error interrupt flag

This bit is set by hardware, only when CAEIE is enabled. This bit is cleared by software by writing it to ‘0’ or masked when CAEIE is ‘0’.

0: No capture error has been detected

1: A capture error has been detected

Bit 1 **COF:** Comparer interrupt flag

This field is set by hardware, only when COIE is enabled. This bit is cleared by software by writing it to ‘0’ or masked when COIE is ‘0’.

0: No counter compare occurred

1: A counter compare has occurred

Bit 0 **CAF:** Capture trigger interrupt flag

This field is set by hardware, only when CAIE is enabled. This bit is cleared by software by writing it to ‘0’ or masked when CAIE is ‘0’.

0: No capture update occurred

1: A capture update has occurred

12.8.47 RCC audio synchronization counter register (RCC_ASCNTR)

Address offset: 0x1CC

Reset value: 0x0000 0000

Access to this register can be protected by GTZC_TZSC SAI1SEC and RCC SPRIV or RCC NSPRIV.

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CNT[19:0]:** Counter value

This field is set by hardware.

CNT[19:0] is the counter value at the time this register is read.

12.8.48 RCC audio synchronization auto-reautoreload register (RCC_ASARR)

Address offset: 0x1D0

Reset value: 0x0008 0000

Access to this register can be protected by GTZC_TZSC SAI1SEC and RCC SPRIV or RCC NSPRIV.

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AR[19:0]:** Auto-reload value
This field is set by software.
This field can only be written by software when the counter is disabled (RCC_ASCR.CEN = 0)
CA[19:0] is the counter auto-reload value at which to restart the audio synchronization counter from value 0. It defines the counter period.

12.8.49 RCC audio synchronization capture register (RCC_ASCAR)

Address offset: 0x1D4

Reset value: 0x0000 0000

Access to this register can be protected by GTZC_TZSC SAI1SEC and RCC SPRIV or RCC NSPRIV.

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

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:0 **CA[26:0]:** Capture value
This field is set by hardware.
CA[26:20] is the capture period counter value loaded on the trigger event.
CA[19:0] is the audio synchronization counter value loaded on the trigger event.

12.8.50 RCC audio synchronization compare register (RCC_ASCOR)

Address offset: 0x1D8

Reset value: 0x0000 0000

Access to this register can be protected by GTZC_TZSC SAI1SEC and RCC SPRIV or RCC NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CO[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CO[19:0]:** Compare value

This field is set by software.

CO[19:0] is the value to be compared to the audio synchronization counter to generate an compare interrupt event.

12.8.51 RCC clock configuration register 2 (RCC_CFGR4)

Address offset: 0x200

Reset value: 0x0000 0010

Access: word, half-word half-word, and byte access

1 or 2 wait states are inserted only if the access occurs during clock source switch.

Access to this register can be protected by RCC PRESCSEC and RCC SPRIV or RCC NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDIV5	Res.	HPRE5[2:0]		
											rw		rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **HDIV5**: AHB5 for 2.4 GHz RADIO divider when SWS select HSI16 or HSE32
Set and reset by software.
Set to 1 by hardware when entering Stop 1, 2, and 3 mode.
– When SYSCLK source indicated by SWS is HSI16 or HSE32: HDIV5 is taken into account
– When SYSCLK source indicated by SWS is PLL1: HDIV5 is taken not taken into account

Caution: Depending on the device voltage range, the software must set this bit correctly to ensure that the AHB5 frequency for the 2.4 GHz RADIO does not exceed the maximum allowed frequency (for more details, refer to *Table 109*). After a write operation to this bit and before decreasing the voltage range, this register must be read to be sure that the new value is taken into account.

0: hclk5 = SYSCLK not divided
1: hclk5 = SYSCLK divided by 2

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **HPRE5[2:0]**: AHB5 for the 2.4 GHz RADIO prescaler when SWS select PLL1
Set and cleared by software to control the division factor of the AHB5 clock (hclk5).
Must not be changed when SYSCLK source indicated by SWS is PLL1.
– When SYSCLK source indicated by SWS is not PLL1: HPRE5 is not taken into account.
– When SYSCLK source indicated by SWS is PLL1: HPRE5 is taken into account, from the moment the system clock switch occurs

Caution: Depending on the device voltage range, the software must set these bits correctly to ensure that the AHB5 frequency for the 2.4 GHz RADIO does not exceed the maximum allowed frequency (for more details, refer to *Table 109: SYSCLK and bus maximum frequency*). After a write operation to these bits and before decreasing the voltage range, this register must be read to be sure that the new value is taken into account.

0xx: hclk5 = SYSCLK not divided
100: hclk5 = SYSCLK divided by 2
101: hclk5 = SYSCLK divided by 3
110: hclk5 = SYSCLK divided by 4
111: hclk5 = SYSCLK divided by 6

12.8.52 RCC RADIO peripheral clock enable register (RCC_RADIOENR)

Address offset: 0x208

Reset value: 0x0000 0000

Access: no wait state; word, half-word half-word, and byte access

Access to this register can be protected by GTZC_TZSC RADIOSEC and RCC SPRIV or NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RADIOCLKRDY	STRADIOCLKON
														r	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BBCLKEN	Res.
														rw	

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **RADIOCLKRDY**: 2.4 GHz RADIO bus clock ready.

Set and cleared by hardware to indicate that the 2.4 GHz RADIO bus clock is ready and the 2.4 GHz RADIO registers can be accessed.

0: 2.4 GHz RADIO bus clock not ready

1: 2.4 GHz RADIO bus clock ready

Note: Once both RADIOEN and STRADIOCLKON are cleared, RADIOCLKRDY goes low after three hclk5 clock cycles.

Bit 16 **STRADIOCLKON**: 2.4 GHz RADIO bus clock enable and HSE32 oscillator enable by 2.4 GHz RADIO sleep timer wake-up event

Set by hardware on a 2.4 GHz RADIO sleep timer wake-up event.

Cleared by software writing zero to this bit.

0: 2.4 GHz RADIO bus clock and HSE32 oscillator not requested by 2.4 GHz RADIO sleep timer wake-up event.

1: 2.4 GHz RADIO bus clock and HSE32 oscillator enabled by 2.4 GHz RADIO sleep timer wake-up event

Note: Before accessing the 2.4 GHz RADIO registers the RADIOCLKRDY bit must be checked.

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **BBCLKEN**: 2.4 GHz RADIO baseband kernel clock (aclk) enable

Set and cleared by software.

0: 2.4 GHz RADIO baseband kernel clock disabled

1: 2.4 GHz RADIO baseband kernel clock enabled

Note: The HSE32 oscillator needs to be enabled by either HSEON or STRADIOCLKON.

Bit 0 Reserved, must be kept at reset value.

12.8.53 RCC external clock sources calibration register 1(RCC_ECSCR1)

Address offset: 0x210

Power-on reset value: 0x14A0 8000

Reset value not effected by exit Standby mode, nor reset from system reset and BORx (x = 1 to 4).

Access: no wait state; word, half-word half-word, and byte access

Access to this register can be protected by RCC HSESEC and RCC SPRIV or RCC NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	HSERDYTRIM[6:0]							HSESFSEN	Res.	HSETRIM[5:0]					
	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XORDYTRIM	HSELDOSEL	Res.	Res.	HSELDOTRIM[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw			rw	rw	rw	rw								

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **HSERDYTRIM[6:0]**: HSE32 clock ready trimming
HSE clock ready delay value.
HSERDY delay = N * 8 µs. With minimum delay = 160 µs.
Writing values smaller than 0x14 are discarded.
In all cases HSERDY is qualified by hardware XO32 clock ready.

Bit 23 **HSESFSEN**: HSE32 fast startup enable
0: HSE fast startup disabled.
1: HSE fast startup enabled.

Bit 22 Reserved, must be kept at reset value.

Bits 21:16 **HSETRIM[5:0]**: HSE32 clock trimming
These bits provide user-programmable capacitor trimming value. It can be programmed to adjust the HSE32 oscillator frequency.

Bit 15 **XORDYTRIM**: XO32 clock ready trimming
0: XO32 ready after 512 cycles
1: XO32 ready after 1024 cycles

Bit 14 **HSELDOSEL**: HSE32 LDO trim selection
0: HSE LDO trim selected from 2.4 GHz RADIO
1: HSE LDO trim selected from HSELDOTRIM

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:8 **HSELDOTRIM[3:0]**: HSE32 LDO voltage level trimming
This value is only used when HSELDOSEL = 1.
00: lowest voltage level
01: low mid voltage level
10: high mid voltage level
11: high voltage level

Bits 7:0 Reserved, must be kept at reset value.

12.8.54 RCC LSI key register (RCC_LSIKEYR)

Address offset: 0x220

Reset value: 0x0000 0000

Access: word, half-word half-word, and byte access

Access to this register can be protected by RCC LSISEC and RCC SPRIV or RCC NSPRIV.

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **KEY[7:0]**: LSI1PREDIV write access enable key

Set and cleared by software to allow write access to LSI1PREDIV even when LSI1 is enabled and ready.

0x4D: write access to LIS1PREDIV granted at all times.

other: write access to LSI1PREDIV disabled when LSI1 is enabled and ready.

12.8.55 RCC register map

Table 117. RCC register map and reset values

Offset	Register	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
0x000	RCC_CR	Res.	Res.	Res.	Res.	Res.	Res.	PLL1RDY	PLL1ON	Res.	Res.	Res.	HSEPRE	HSECSSON	Res.	HSERDY	HSEON	Res.	Res.	Res.	Res.	Res.	HSIRDY	HSIKERON	HSION	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value							0	0				0	0		0	0						0	0	0								
0x004 to 0x00C	Reserved	Reserved																															
0x010	RCC_ICSCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSITRIM[4:0]					Res.	Res.	Res.	Res.	HSICAL[11:0]											
	Reset value												1	0	0	0	0					X	X	X	X	X	X	X	X	X	X	X	X
0x014 to 0x0018	Reserved	Reserved																															
0x01C	RCC_CFGR1	Res.	MCOPRE[2:0]			MCOSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWS[1:0]		SW[1:0]	
	Reset value		0	0	0	0	0	0	0																					0	0	0	0
0x020	RCC_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPRE2[2:0]			Res.	PPRE1[2:0]			Res.	HPRE[2:0]		
	Reset value																						0	0	0		0	0	0		0	0	0
0x024	RCC_CFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPRE7[2:0]			Res.	Res.	Res.	Res.
	Reset value																										0	0	0				
0x028	RCC_ PLL1CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL1RCLKPRERDY	PLL1RCLKPRESTEP	PLL1RCLKPRE	Res.	PLL1REN	PLL1QEN	PLL1PEN	Res.	Res.	Res.	Res.	Res.	PLL1M[2:0]			Res.	Res.	PLL1FRACEN	PLL1RGE[1:0]		PLL1SRC[1:0]		
	Reset value										0	0	0		0	0	0						0	0	0			0	0	0	0	0	0
0x02C to 0x030	Reserved	Reserved																															
0x034	RCC_ PLL1DIVR	Res.	PLL1R[6:0]							Res.	PLL1Q[6:0]							PLL1P[6:0]							PLL1N[8:0]								
	Reset value		0	0	1	0	0	0	1		0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x038	RCC_ PLL1FRACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL1FRACN[12:0]													Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x03C to 0x04C	Reserved	Reserved																															

Table 117. RCC register map and reset values (continued)

Offset	Register	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
0x050	RCC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSI2RDYIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL1RDYIE	Res.	HSERDYIE	HSIRDYIE	Res.	LSERDYIE	LSI1RDYIE
	Reset value																0										0		0	0		0	0
0x054	RCC_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSI2RDYF	Res.	Res.	Res.	Res.	Res.	HSECSSF	Res.	Res.	Res.	PLL1RDYF	Res.	HSERDYF	HSIRDYF	Res.	LSERDYF	LSI1RDYF
	Reset value																0						0				0		0	0		0	0
0x058	RCC_CICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSI2RDYC	Res.	Res.	Res.	Res.	Res.	HSECSSC	Res.	Res.	Res.	PLL1RDYC	Res.	HSERDYC	HSIRDYC	Res.	LSERDYC	LSI1RDYC
	Reset value																0						0				0		0	0		0	0
0x05C	Reserved	Reserved																															
0x060	RCC_ AHB1RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPDMA1RST
	Reset value																				0												0
0x064	RCC_ AHB2RSTR	Res.	Res.	XSPI1RST	Res.	Res.	Res.	Res.	Res.	OTFDEC1RST	Res.	PKARST	Res.	Res.	RNGRST	HASHRST	AESRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOHRST	Res.	Res.	Res.	Res.	GPIOCRST	GPIOBRST	GPIOARST
	Reset value			0						0		0			0	0	0									0					0	0	0
0x068	Reserved	Reserved																															
0x06C	RCC_ AHB4RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4RST	Res.	Res.	Res.	Res.	Res.
	Reset value																											0					
0x070	RCC_ AHB5RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTACONVRST	RADIORST
	Reset value																															0	0
0x074	RCC_ APB1RSTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2RST
	Reset value											0																					0
0x078	RCC_ APB1RSTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2RST	Res.	Res.	Res.	Res.	Res.
	Reset value																											0					

Table 117. RCC register map and reset values (continued)

Offset	Register	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
0x07C	RCC_ APB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBRST	Res.	Res.	SAI1RST	Res.	Res.	TIM17RST	TIM16RST	Res.	Res.	USART1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0			0			0	0			0														
0x080	RCC_ APB7RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1RST	Res.	Res.	Res.	I2C3RST	LPUART1RST	SPI3RST	Res.	Res.	Res.	SYSCFGRST	Res.
	Reset value																					0				0	0	0				0	
0x084	Reserved	Reserved																															
0x088	RCC_ AHB1ENR	SRAM1EN	Res.	Res.	Res.	Res.	Res.	Res.	GTZC1EN	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGEN	Res.	Res.	Res.	Res.	CRCEN	Res.	Res.	Res.	FLASHEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPDMA1EN
	Reset value	1							0							0					0				1								0
0x08C	RCC_ AHB2ENR	Res.	SRAM2EN	XSPI1EN	Res.	Res.	Res.	Res.	Res.	OTFDEC1EN	Res.	PKAEN	Res.	Res.	RNGEN	HASHEN	AESEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOHEN	Res.	Res.	Res.	Res.	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value		0	0						0		0			0	0	0									0					0	0	0
0x090	Reserved	Reserved																															
0x094	RCC_ AHB4ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4EN	Res.	Res.	PWREN	Res.	Res.
	Reset value																											0			0		
0x098	RCC_ AHB5ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTACONVEN	RADIOEN
	Reset value																															0	0
0x09C	RCC_ APB1ENR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2EN
	Reset value											0																					0
0x0A0	RCC_ APB1ENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2EN	Res.	Res.	Res.	Res.	Res.
	Reset value																											0					
0x0A4	RCC_ APB2ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBEN	Res.	Res.	SAI1EN	Res.	Res.	TIM17EN	TIM16EN	Res.	Res.	USART1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0			0			0	0			0														
0x0A8	RCC_ APB7ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1EN	Res.	Res.	Res.	I2C3EN	LPUART1EN	SPI3EN	Res.	Res.	Res.	SYSCFGEN	Res.
	Reset value											0										0				0	0	0				0	

Table 117. RCC register map and reset values (continued)

Offset	Register	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
0x0AC	Reserved	Reserved																															
0x0B0	RCC_ AHB1SMENR	SRAM1SMEN	Res.	ICACHESMEN	Res.	Res.	Res.	Res.	GTZC1SMEN	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGSMEN	Res.	Res.	Res.	Res.	CRCSMEN	Res.	Res.	Res.	FLASHSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPDMA1SMEN
	Reset value	1		1					1							1					1				1								1
0x0B4	RCC_ AHB2SMENR	Res.	SRAM2SMEN	XSPI1SMEN	Res.	Res.	Res.	Res.	Res.	OTFDEC1SMEN	Res.	PKASMEN	Res.	Res.	RNGSMEN	HASHSMEN	AESSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOHSMEN	Res.	Res.	Res.	Res.	GPIOCSMEN	GPIOBSMEN	GPIOASMEN
	Reset value		1	1						1		1			1	1	1									1					1	1	1
0x0B8	Reserved	Reserved																															
0x0BC	RCC_ AHB4SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4SMEN	Res.	Res.	PWRSMEN	Res.	Res.
	Reset value																											1			1		
0x0C0	RCC_ AHB5SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTACONVSMEN	RADIOSMEN
	Reset value																															1	1
0x0C4	RCC_ APB1SMENR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C1SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2SMEN
	Reset value											1																					1
0x0C8	RCC_ APB1SMENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2SMEN	Res.	Res.	Res.	Res.	Res.
	Reset value																											1					
0x0CC	RCC_ APB2SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res	USBSMEN	Res.	Res.	SAI1SMEN	Res.	Res.	TIM17SMEN	TIM16SMEN	Res.	Res.	USART1SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								1			1			1	1			1														
0x0D0	RCC_ APB7SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1SMEN	Res.	Res.	Res.	I2C3SMEN	LPUART1SMEN	SPI3SMEN	Res.	Res.	Res.	SYSCFGSMEN	Res.
	Reset value											1										1				1	1	1				1	
0x0D4 to 0x0DC	Reserved	Reserved																															

Table 117. RCC register map and reset values (continued)

Offset	Register	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
0x0E0	RCC_ CCIPR1	TIMICSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSTICKSEL [1:0]		Res.	Res.	LPTIM2SEL [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	I2C1SEL [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART1SEL [1:0]	
	Reset value	0								0	0			0	0							0	0									0	0
0x0E4	RCC_ CCIPR2	Res.	ASSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGSEL [1:0]		Res.	Res.	Res.	Res.	SAI1SEL [2:0]			Res.	Res.	Res.	Res.	Res.
	Reset value		0																	0	0					0	0	0					
0x0E8	RCC_ CCIPR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADCSEL [2:0]			LPTIM1SEL [1:0]		Res.	Res.	I2C3SEL [1:0]		Res.	SPI3SEL [1:0]		Res.	LPUART1SEL [1:0]	
	Reset value																		0	0	0	0	0			0	0		0	0		0	0
0x0F0	RCC_BDCR1	LSICLK	LSI2RDY	LSI2ON	LSI1PREDIV	LSI1RDY	LSI1ON	LSCOSEL	LSCOEN	LSEGATED	LSIGATED	LSEGATEEN	LSIGATEEN	RADIOSTSEL [1:0]		Res.	BDRST	Res.	LSETRIM [1:0]		LSEGFRDY	LSESYSRDY	Res.	RTCSEL [1:0]		LSESYSEN	LSECSSD	LSECSSON	LSEDRV [1:0]		LSEBYP	LSERDY	LSEON
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0		X	X	0	0		0	0	0	0	0	0	1	0	0	0
0x0F4	RCC_CSR	LPWRRSTF	Res.	IWDGRSTF	SFTRSTF	BORRSTF	PINRSTF	OBLRSTF	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0		0	0	1	1	0		0																							
0x0F8	RCC_BDCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSI2CFG[3:0]				Res.	LSI2MODE[2:0]		
	Reset value																									0	0	0	0		0	0	0
0x0FC to 0x10C	Reserved	Reserved																															
0x110	RCC_ SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RMVFSEC	Res.	Res.	Res.	Res.	PLL1SEC	PRESCSEC	SYSCLKSEC	LSESEC	LSISEC	Res.	HSESEC	HSISEC
	Reset value																				0					0	0	0	0	0		0	0
0x114	RCC_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NSPRIV	SPRIV
	Reset value																															0	0
0x118 to 0x1BC	Reserved	Reserved																															
0x1C0	**RCC_ASCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TPS[6:0]							Res.	PSC[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEN
	Reset value										0	0	0	0	0	0	0		1	0	0	0	0	0	0								0
0x1C4	**RCC_ASIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARIE	CAEIE	COIE	CAIE
	Reset value																													0	0	0	0

Table 117. RCC register map and reset values (continued)

Offset	Register	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
0x1C8	**RCC_ASSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARF	CAEF	COF	CAF
	Reset value																													0	0	0	0
0x1CC	**RCC_ASCNTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1D0	**RCC_ASARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR[19:0]																			
	Reset value													1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1D4	**RCC_ASCAR**	Res.	Res.	Res.	Res.	Res.	CA[26:0]																										
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1D8	**RCC_ASCOR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CO[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1DC to 0x1FC	Reserved	Reserved																															
0x200	RCC_CFGR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDIV5	Res.	HPRE5[2:0]		
	Reset value																												1		0	0	0
0x204	Reserved	Reserved																															
0x208	RCC_ RADIOENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RADIOCLKRDY	STRADIOCLKON	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BBCLKEN	Res.
	Reset value															0	0															0	
0x20C	Reserved	Reserved																															
0x210	RCC_ECSCR1	Res.	HSERDYTRIM[6:0]							HSEFSEN	Res.	HSETRIM[5:0]						XORDYTRIM	HSELDOSEL	Res.	Res.	HSELDOTRIM[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0	0	1	0	1	0	0	1		1	0	0	0	0	0	1	0			0	0	0	0								
0x214 to 0x21C	Reserved	Reserved																															
0x220	**RCC_LSIKEYR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x224 to 0x3FC	Reserved	Reserved																															

Refer to *Section 2.3.2: Memory map and register boundary addresses* for the register boundary addresses.

13 General-purpose I/Os (GPIO)

13.1 GPIO introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), a 16-bit reset register (GPIOx_BRR) and a 32-bit set/reset register (GPIOx_BSRR).

In addition, all GPIOs have a 32-bit locking register (GPIOx_LCKR), two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL), and a secure configuration register (GPIOx_SECCFGR).

13.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input-data to input-data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_BSRR) for bitwise write access to GPIOx_ODR
- Lock mechanism (GPIOx_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions
- High speed at low I/O voltage operation
- TrustZone® security support

13.3 GPIO implementation

Table 118. GPIO implementation

GPIO features	GPIOA	GPIOB	GPIOC	GPIOH
STM32WBA25xx GPIO port pins	[15:5,2:1]	[15,12,9:8,6:2]	[15:13]	[3]
STM32WBA23xx GPIO port pins	[15:5,2:0]			

13.4 GPIO functional description

Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be configured individually by software in several modes:

- Input floating
- Input pull-up
- Input pull-down
- Analog
- Analog pull-down
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable; however, the I/O port registers must be accessed as 32-bit words, half-words, or bytes. The GPIOx_BSRR and GPIOx_BRR registers allow atomic read/modify accesses to any of the GPIOx_ODR registers. In this way, there is no risk of an IRQ occurring between the read and modify access operations.

GPIO configuration is only available in Run and Stop modes, it keeps the I/O port in its defined state. In Standby mode, the GPIO configuration is lost. To keep the I/O port input definition and output levels, a GPIO standby retention can be enabled in the PWR (*Section 11.7.10: PWR Standby mode*).

Figure 42 shows the basic structures of a 3 or 5 V-tolerant GPIO (TT or FT). *Table 119* gives the possible port bit configurations.

Figure 42. Structure of 3 or 5 V-tolerant GPIO (TT or FT)

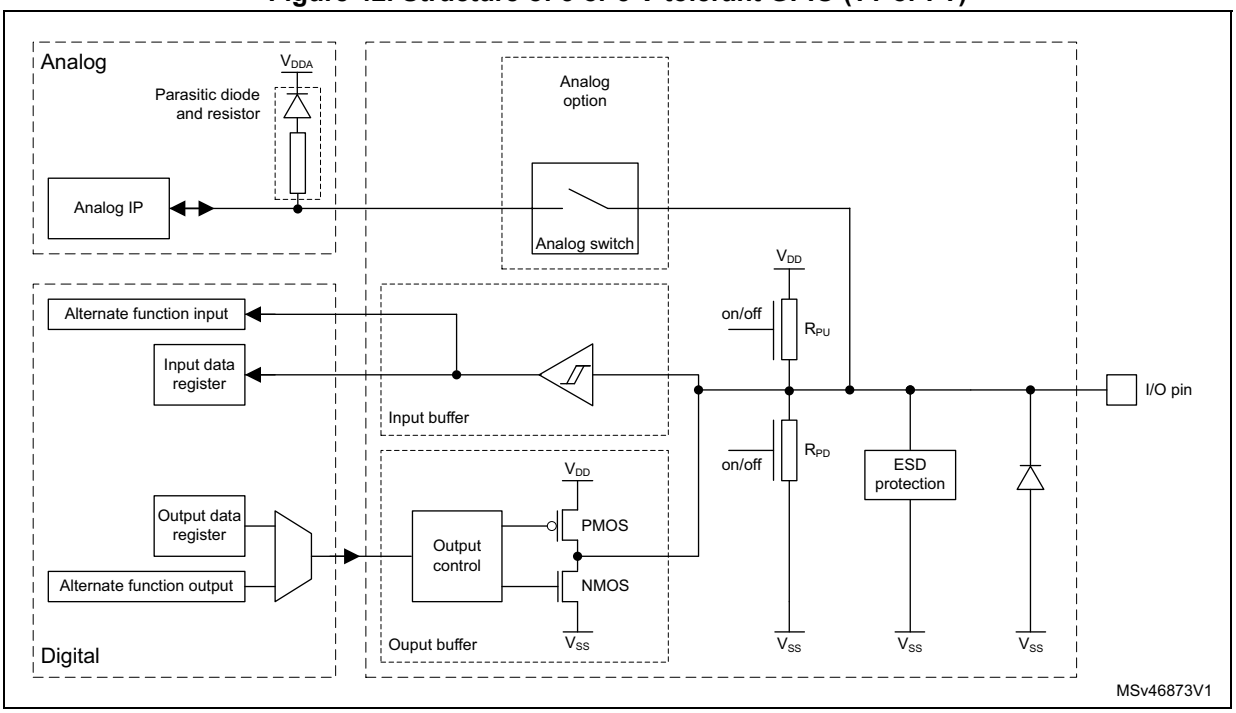


Table 119. Port bit configuration(1)

MODE(i) [1:0]	OTYPE(i)	OSPEED(i) [1:0]		PUPD(i) [1:0]		I/O configuration	
01	0	SPEED [1:0]		0	0	GP output	PP
	0			0	1	GP output	PP + PU
	0			1	0	GP output	PP + PD
	0			1	1	Reserved	
	1			0	0	GP output	OD
	1			0	1	GP output	OD + PU
	1			1	0	GP output	OD + PD
	1			1	1	Reserved (GP output OD)	
10	0	SPEED [1:0]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0	Input/output	Analog PD
	x	x	x	1	1	Reserved	

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

13.4.1 GPIO general-purpose I/O

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in analog mode.

The debug pins are in AF pull-up/pull-down after reset:

- PA15: JTDI in pull-up
- PA14: JTCK/SWCLK in pull-down
- PA13: JTMS/SWDIO in pull-up
- PB4: NJTRST in pull-up
- PB3: JTDO/TRACESWO in floating state, no pull-up/pull-down

PH3/BOOT0 is in input mode during the reset until at least the end of the option byte loading phase (see *Section 13.4.17*).

When the pin is configured as output, the value written in the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven; high level is high-Z).

The input data register (GPIOx_IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors that can be activated or deactivated depending on the value in the GPIOx_PUPDR register.

13.4.2 GPIO pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. This way, there is no conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to 16 alternate function inputs (AF0 to AF15) that can be configured through the GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to 15) registers:

- After reset, the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through the GPIOx_MODER register.
- The specific alternate function assignments for each pin are detailed in the device datasheet.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

For secure peripherals, the GPIO pin must also be secure to use the secure peripheral alternate function.

To use an I/O in a given configuration, the user must proceed as follows:

- **Debug function**: after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host.
- **GPIO**: configure the desired I/O as output, input, or analog in the GPIOx_MODER register.
- **Peripheral alternate function**:
  - Connect the I/O to the desired alternate function in one of the GPIOx_AFRL or GPIOx_AFRH register.
  - Select the type, pull-up/pull-down, and output speed via the GPIOx_OTYPER, GPIOx_PUPDR, and GPIOx_OSPEEDER registers respectively.
  - Configure the desired I/O as an alternate function in the GPIOx_MODER register.
- **Additional functions:**
  - For the ADC, configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC registers.
  - For the additional functions like RTC, TAMP, WKUP, and LSE configure the required function in the related peripheral register. These functions have priority over the configuration in the GPIO registers.

Refer to the “Alternate function mapping” table in the device datasheet for the detailed mapping of the alternate function I/O pins.

13.4.3 GPIO port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) to configure up to 16 I/Os. The GPIOx_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIOx_OTYPER and GPIOx_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed. The GPIOx_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

13.4.4 GPIO port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (*GPIO port A input data register (GPIOA_IDR)* and *GPIO port A output data register (GPIOA_ODR)*).

GPIOx_ODR stores the data to be output, it is read/write accessible. The data that are input through the I/O are stored in the input data register (GPIOx_IDR), which is a read-only register.

13.4.5 GPIO data bitwise handling

The bit set reset register (GPIOx_BSRR) is a 32-bit register that allows the application to set and reset each individual bit in the output data register (GPIOx_ODR). This register is twice the size of GPIOx_ODR.

Two control bits in GPIOx_BSRR, namely BSy and BRy correspond to each bit in GPIOx_ODR. When set, BSy sets the corresponding ODRy bit. When BRy is set, it resets the corresponding ODRy bit.

Clearing a bit in GPIOx_BSRR does not affect the corresponding bit in GPIOx_ODR. If there is an attempt to set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using the GPIOx_BSRR register to change the values of individual bits in GPIOx_ODR is a “one-shot” effect that does not lock the GPIOx_ODR bits. The GPIOx_ODR bits can always be accessed directly. The GPIOx_BSRR register provides a way of performing atomic bitwise handling.

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level: one or more bits can be modified in a single atomic AHB write access.

Individual bits in GPIOx_ODR can also be reset in a single atomic AHB write to GPIOx_BRR.

13.4.6 GPIO locking mechanism

The GPIO control registers can be frozen by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL, and GPIOx_AFRH.

To write the GPIOx_LCKR register, a specific write/read sequence must be applied. When the right LOCK sequence is applied to bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence is applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL, and GPIOx_AFRH.

The LOCK sequence can only be performed using (32-bit) word access to the GPIOx_LCKR register because GPIOx_LCKR bit 16 must be set at the same time as bits [15:0].

13.4.7 GPIO alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, the user can connect an alternate function to some other pin as required by the application.

This means that a number of possible peripheral functions are multiplexed on each GPIO using the GPIOx_AFRL and GPIOx_AFRH alternate function registers. The application can thus select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin, refer to the device datasheet.

13.4.8 GPIO external interrupt/wake-up lines

All ports have external interrupt capability. To use external interrupt lines, the port can be configured in input, output, or alternate function mode (the port must not be configured in analog mode). Refer to *Section 18: Extended interrupts and event controller (EXTI)*.

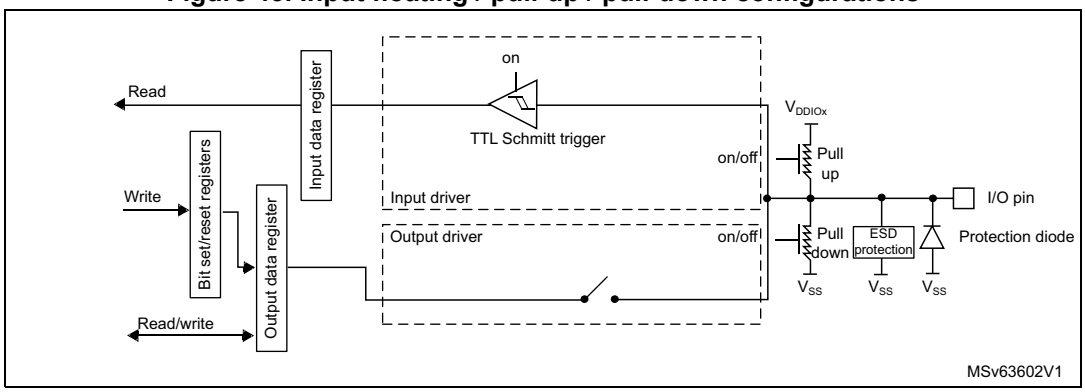
13.4.9 GPIO input configuration

When the I/O port is programmed as input:

- The output buffer is disabled.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- Read access to the input data register provides the I/O state.

Figure 43 shows the input configuration of the I/O port bit.

Figure 43. Input floating / pull-up / pull-down configurations



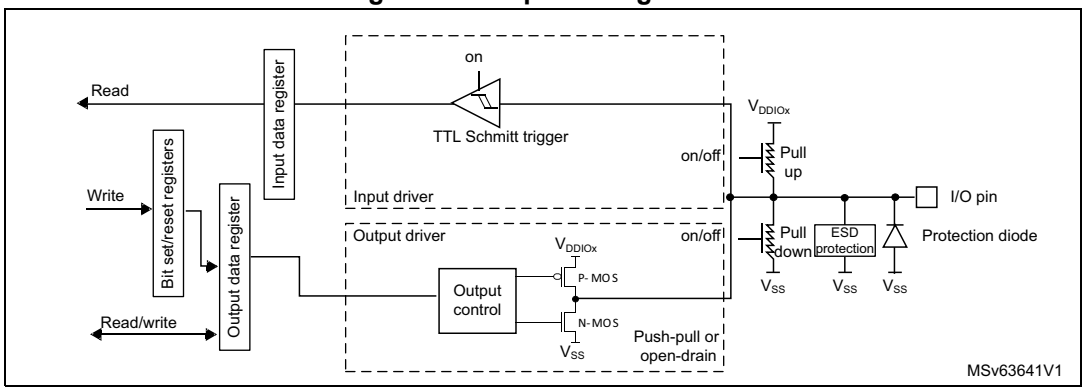
13.4.10 GPIO output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open-drain mode: a 0 in the output register activates the N-MOS, whereas a 1 in the output register leaves the port in high-Z (the P-MOS is never activated).
  - Push-pull mode: a 0 in the output register activates the N-MOS whereas a 1 in the output register activates the P-MOS.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- Read access to the input data register gets the I/O state.
- Read access to the output data register gets the last written value.

Figure 44 shows the output configuration of the I/O port bit.

Figure 44. Output configuration



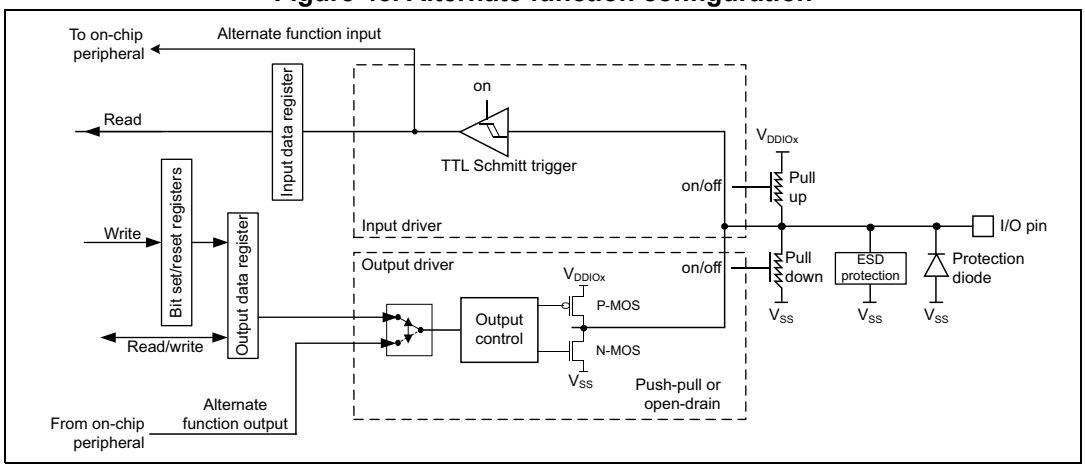
13.4.11 GPIO alternate function configuration

When the I/O port is programmed as alternate function:

- The output buffer can be configured in open-drain or push-pull mode.
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data).
- The Schmitt trigger input is activated.
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- Read access to the input data register gets the I/O state.

Figure 45 shows the alternate function configuration of the I/O port bit.

Figure 45. Alternate function configuration



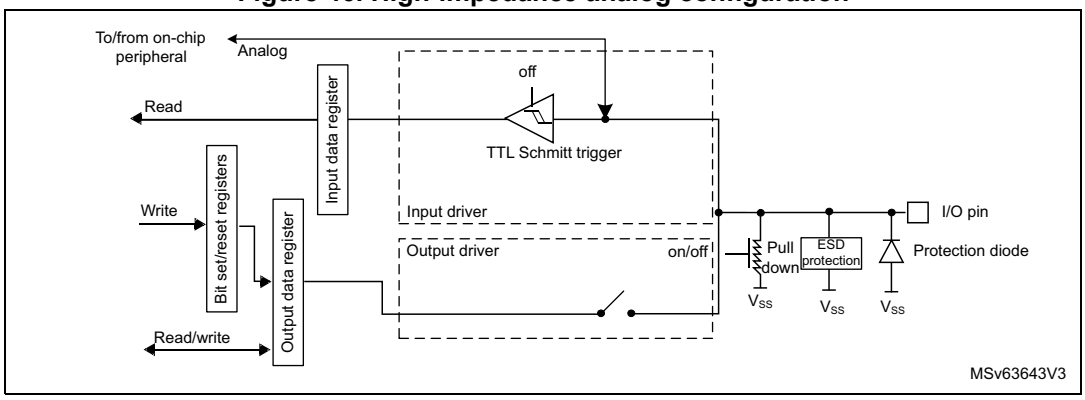
13.4.12 GPIO analog configuration

When the I/O port is programmed as analog configuration:

- The output buffer is disabled.
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-down resistor is activated depending on the value in the GPIOx_PUPDR register. The weak pull-up resistor is disabled by hardware.
- Read access to the input data register gets the value 0.

Figure 46 shows the high-Z, analog-input configuration of the I/O port bits.

Figure 46. High-impedance analog configuration



13.4.13 GPIO compensation cell

The I/O commutation slew rate (t_{fall}/t_{rise}) can be adapted by software depending on process, voltage and temperatures conditions, to reduce the I/O noise on the power supply. Refer to *Section 14: System configuration controller (SYSCFG)* for more details.

13.4.14 GPIO Standby retention

The I/O state can be retained in Standby mode. This is configured in the PWR.

13.4.15 GPIO using the LSE oscillator pins as GPIOs

When the LSE oscillator is switched off (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the LSE oscillator is switched on (by setting the LSEON bit in the RCC_BDCR register), the oscillator takes control of its associated pins and the GPIO configuration of these pins has no effect.

When the oscillator is configured in a user external clock mode, only the pin is reserved for clock input, and the OSC32_OUT pin can still be used as normal GPIO.

13.4.16 GPIO using GPIO pins with RTC

The PC13/PC14/PC15 GPIO functionality is lost when the Core domain is powered off (when the device enters Standby mode). In this case, if their GPIO configuration is not

bypassed by the RTC configuration, these pins are, when not retained in Standby mode, set in analog input mode.

For details about I/O control by the RTC, refer to *Section 33.3: RTC functional description*.

13.4.17 GPIO using PH3 as GPIO

PH3 may be used as boot pin (BOOT0) or as a GPIO. Depending on the nSWBOOT0 user option bit in the FLASH_OPTR register, PH3 switches from input mode to analog input mode. This happens:

- After the option byte loading phase if nSWBOOT0 = 1
- After reset if nSWBOOT0 = 0

13.4.18 GPIO TrustZone® security

The TrustZone® security is activated by the TZEN user option bit in the FLASH_OPTR register. When TrustZone® is enabled (TZEN = 1), each I/O pin of the GPIO port can be individually configured as secure through the GPIOx_SECCFGR register.

When the selected I/O pin is configured as secure, its corresponding configuration bits for alternate function, mode selection, and I/O data are secured from nonsecure access operations. In the case of a nonsecure access operation, these bits are RAZ/WI.

The I/Os with peripheral functions are also conditioned by the peripheral security configuration (see *Section 5: Global TrustZone® controller (GTZC)* for more details):

- For peripherals for which the I/O pin selection is done through alternate function registers: if the peripheral is configured as secure, it cannot be connected to a nonsecure I/O pin. If this is not respected, the input data to the secure peripheral is forced to 0 (the I/O input pin value is ignored) and the output pin value is forced to 0, thus avoiding any secure information leaks through nonsecure I/Os.
- For I/Os with analog switches, directly controlled by peripherals (such as ADC): if the I/O is secure, the I/O analog switch cannot be controlled by a nonsecure peripheral. If this is not respected, the switch remains open. This prevents the redirection of secure data to a nonsecure peripheral or I/O through analog path. Refer to *Section 4: System security* for more details.
- Some of the paths between I/O “additional functions” and peripherals are not blocked if the I/O is secure and the peripheral is nonsecure. Therefore, it is recommended that those peripherals be configured as secure even when not used by the application. Refer to *Section 4: System security* for the list of concerned peripherals. When the path has a security control, it follows the same rule as I/O selection through alternate functions.

Refer to the device pins definition table in the datasheet for more information about peripheral alternate functions and additional function mapping.

After reset, all GPIO ports are secure.

Table 120 gives a summary of the I/O port secured bits following the security configuration bit in the GPIOx_SECCFGR register. When the I/O bit port is configured as secure:

- Secured bits: read and write operations are only allowed by secure access. Nonsecure read or write access operations on secured bits are RAZ/WI. No illegal access event is generated.
- Nonsecure bits: no restriction. Read and write operations are allowed by both secure and nonsecure access operations.

When TrustZone® security is disabled (TZEN = 0), all registers bits are nonsecure. The GPIOx_SECCFGR register is RAZ/WI.

Table 120. GPIO secured bits

Secure configuration bit	Secured bit	Register name	Nonsecure access on secure bits
SECy = 1 in GPIOx_SECCFGR(1)	MODEy[1:0]	GPIOx_MODER	RAZ/WI
	OTy	GPIOx_OTYPER	
	OSPEEDy[1:0]	GPIOx_OSPEEDR	
	PUPDy[1:0]	GPIOx_PUPDR	
	IDy	GPIOx_IDR	
	ODy	GPIOx_ODR	
	BSy and BRy	GPIOx_BSRR	
	LCKy	GPIOx_LCKR	
	BRy	GPIOx_BRR	
	AFSELy[3:0]	GPIOx_AFRH	
		GPIOx_AFRL	

1. x = port index, y = port pin index, for values see *Table 118: GPIO implementation*.

As soon as at least one function is configured as secure, the GPIO reset and clock control bits in the RCC are also secured.

13.4.19 GPIO privileged and unprivileged modes

All GPIO registers can be read and written by privileged and unprivileged access, whatever the security state (secure or nonsecure).

13.5 GPIO port A registers

This section gives a detailed description of the GPIO port A registers.

The peripheral registers can be written in word, half-word, or byte mode.

13.5.1 GPIO port A mode register (GPIOA_MODER)

Address offset: 0x000

Reset value: 0xABFF FC3F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		Res.	Res.	Res.	Res.	MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:30 **MODE15[1:0]:** Port configuration I/O pin 15
These bits are written by software to configure the I/O mode.
Access can be protected by GPIOA SEC15.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

Bits 29:28 **MODE14[1:0]:** Port configuration I/O pin 14

Bits 27:26 **MODE13[1:0]:** Port configuration I/O pin 13

Bits 25:24 **MODE12[1:0]:** Port configuration I/O pin 12

Bits 23:22 **MODE11[1:0]:** Port configuration I/O pin 11

Bits 21:20 **MODE10[1:0]:** Port configuration I/O pin 10

Bits 19:18 **MODE9[1:0]:** Port configuration I/O pin 9

Bits 17:16 **MODE8[1:0]:** Port configuration I/O pin 8

Bits 15:14 **MODE7[1:0]:** Port configuration I/O pin 7

Bits 13:12 **MODE6[1:0]:** Port configuration I/O pin 6

Bits 11:10 **MODE5[1:0]:** Port configuration I/O pin 5

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **MODE2[1:0]:** Port configuration I/O pin 2

Bits 3:2 **MODE1[1:0]:** Port configuration I/O pin 1

Bits 1:0 **MODE0[1:0]:** Port configuration I/O pin 0

13.5.2 GPIO port A output type register (GPIOA_OTYPER)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	Res.	Res.	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:5 **OT[15:5]:** Port configuration I/O pin y (y = 15 to 5)
These bits are written by software to configure the I/O output type.
Access can be protected by GPIOA SECy.
0: Output push-pull (reset state)
1: Output open-drain

Bits 4:3 Reserved, must be kept at reset value.

Bits 2:0 **OT[2:0]:** Port configuration I/O pin y (y = 2 to 0)

13.5.3 GPIO port A output speed register (GPIOA_OSPEEDR)

Address offset: 0x008

Reset value: 0x0800 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		Res.	Res.	Res.	Res.	OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
rw	rw	rw	rw	rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:30 **OSPEED15[1:0]**: Port configuration I/O pin 15
These bits are written by software to configure the I/O output speed.
Access can be protected by GPIOA SEC15.
00: Low speed
01: Medium speed
10: High speed
11: Reserved

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

Bits 29:28 **OSPEED14[1:0]:** Port configuration I/O pin 14

Bits 27:26 **OSPEED13[1:0]:** Port configuration I/O pin 13

Bits 25:24 **OSPEED12[1:0]:** Port configuration I/O pin 12

Bits 23:22 **OSPEED11[1:0]:** Port configuration I/O pin 11

Bits 21:20 **OSPEED10[1:0]:** Port configuration I/O pin 10

Bits 19:18 **OSPEED9[1:0]:** Port configuration I/O pin 9

Bits 17:16 **OSPEED8[1:0]:** Port configuration I/O pin 8

Bits 15:14 **OSPEED7[1:0]:** Port configuration I/O pin 7

Bits 13:12 **OSPEED6[1:0]:** Port configuration I/O pin 6

Bits 11:10 **OSPEED5[1:0]:** Port configuration I/O pin 5

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **OSPEED2[1:0]:** Port configuration I/O pin 2

Bits 3:2 **OSPEED1[1:0]:** Port configuration I/O pin 1

Bits 1:0 **OSPEED0[1:0]:** Port configuration I/O pin 0

13.5.4 GPIO port A pull-up/pull-down register (GPIOA_PUPDR)

Address offset: 0x00C

Reset value: 0x6400 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		Res.	Res.	Res.	Res.	PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:30 **PUPD15[1:0]:** Port configuration I/O pin 15

These bits are written by software to configure the I/O pull-up or pull-down.
Access can be protected by GPIOA SEC15.
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

Bits 29:28 **PUPD14[1:0]:** Port configuration I/O pin 14

Bits 27:26 **PUPD13[1:0]:** Port configuration I/O pin 13

Bits 25:24 **PUPD12[1:0]:** Port configuration I/O pin 12

Bits 23:22 **PUPD11[1:0]:** Port configuration I/O pin 11

Bits 21:20 **PUPD10[1:0]:** Port configuration I/O pin 10

Bits 19:18 **PUPD9[1:0]:** Port configuration I/O pin 9

Bits 17:16 **PUPD8[1:0]:** Port configuration I/O pin 8

Bits 15:14 **PUPD7[1:0]:** Port configuration I/O pin 7

Bits 13:12 **PUPD6[1:0]:** Port configuration I/O pin 6

Bits 11:10 **PUPD5[1:0]:** Port configuration I/O pin 5

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **PUPD2[1:0]:** Port configuration I/O pin 2

Bits 3:2 **PUPD1[1:0]:** Port configuration I/O pin 1

Bits 1:0 **PUPD0[1:0]:** Port configuration I/O pin 0

13.5.5 GPIO port A input data register (GPIOA_IDR)

Address offset: 0x010

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	Res.	Res.	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r			r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:5 **ID[15:5]:** Port input data I/O pin y (y = 15 to 5)
These bits are read-only. They contain the input value of the corresponding I/O port.
Access can be protected by GPIOA SECy.

Bits 4:3 Reserved, must be kept at reset value.

Bits 2:0 **ID[2:0]:** Port input data I/O pin y (y = 2 to 0)

13.5.6 GPIO port A output data register (GPIOA_ODR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	Res.	Res.	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:5 **OD[15:5]:** Port output data I/O pin y (y = 15 to 5)
These bits can be read and written by software.
Access can be protected by GPIOA SECy.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOA_BSRR or GPIOA_BRR registers.

Bits 4:3 Reserved, must be kept at reset value.

Bits 2:0 **OD[2:0]:** Port output data I/O pin y (y = 2 to 0)

13.5.7 GPIO port A bit set/reset register (GPIOA_BSRR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	Res.	Res.	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w			w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	Res.	Res.	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w			w	w	w

Bits 31:21 **BR[15:5]:** Port reset I/O pin y (y = 15 to 5)
These bits are write-only. A read to these bits returns the value 0.
Access can be protected by GPIOA SECy.
0: No action on the corresponding ODy bit
1: Resets the corresponding ODy bit
Note: If both BSy and BRy are set, BSy has priority.

Bits 20:19 Reserved, must be kept at reset value.

Bits 18:16 **BR[2:0]:** Port reset I/O pin y (y = 2 to 0)

Bits 15:5 **BS[15:5]:** Port set I/O pin y (y = 15 to 5)
These bits are write-only. A read to these bits returns the value 0.
Access can be protected by GPIOA SECy.
0: No action on the corresponding ODy bit
1: Sets the corresponding ODy bit

Bits 4:3 Reserved, must be kept at reset value.

Bits 2:0 **BS[2:0]:** Port set I/O pin y (y = 2 to 0)

13.5.8 GPIO port A configuration lock register (GPIOA_LCKR)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOA_LCKR register. Only (32-bit) word access is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	Res.	Res.	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.
Access can be protected by any GPIOA SECy.
0: Port configuration lock key not active
1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.
- LOCK key write sequence:
WR LCKR[16] = 1 + LCKR[15:0]
WR LCKR[16] = 0 + LCKR[15:0]
WR LCKR[16] = 1 + LCKR[15:0]
- LOCK key read
RD LCKR[16] = 1 (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCKR[15:0] must not change.
Any error in the lock sequence aborts the LOCK.
After the first LOCK sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:5 **LCK[15:5]:** Port lock I/O pin y (y = 15 to 5)

These bits are read/write but can only be written when the LCKK bit is 0
Access can be protected by GPIOA SECy.
0: Port configuration not locked
1: Port configuration locked

Bits 4:3 Reserved, must be kept at reset value.

Bits 2:0 **LCK[2:0]:** Port lock I/O pin y (y = 2 to 0)

13.5.9 GPIO port A alternate function low register (GPIOA_AFRL)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **AFSEL7[3:0]:** Alternate function selection for port I/O pin 7
These bits are written by software to configure alternate function I/Os.
Access can be protected by GPIOA SEC7.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

Bits 27:24 **AFSEL6[3:0]:** Alternate function selection for port I/O pin 6

Bits 23:20 **AFSEL5[3:0]:** Alternate function selection for port I/O pin 5

Bits 19:12 Reserved, must be kept at reset value.

Bits 11:8 **AFSEL2[3:0]:** Alternate function selection for port I/O pin 2

Bits 7:4 **AFSEL1[3:0]:** Alternate function selection for port I/O pin 1

Bits 3:0 **AFSEL0[3:0]:** Alternate function selection for port I/O pin 0

13.5.10 GPIO port A alternate function high register (GPIOA_AFRH)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **AFSEL15[3:0]:** Alternate function selection for port I/O pin 15
These bits are written by software to configure alternate function I/Os.
Access can be protected by GPIOA SEC15.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

Bits 27:24 **AFSEL14[3:0]:** Alternate function selection for port I/O pin 14

Bits 23:20 **AFSEL13[3:0]:** Alternate function selection for port I/O pin 13

Bits 19:16 **AFSEL12[3:0]:** Alternate function selection for port I/O pin 12

Bits 15:12 **AFSEL11[3:0]:** Alternate function selection for port I/O pin 11

Bits 11:8 **AFSEL10[3:0]:** Alternate function selection for port I/O pin 10

Bits 7:4 **AFSEL9[3:0]:** Alternate function selection for port I/O pin 9

Bits 3:0 **AFSEL8[3:0]:** Alternate function selection for port I/O pin 8

13.5.11 GPIO port A bit reset register (GPIOA_BRR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	Res.	Res.	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w			w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:5 **BR[15:5]:** Port reset I/O pin y (y = 15 to 5)
These bits are write-only. A read to these bits returns the value 0.
Access can be protected by GPIOA SECy.
0: No action on the corresponding ODy bit
1: Reset the corresponding ODy bit

Bits 4:3 Reserved, must be kept at reset value.

Bits 2:0 **BR[2:0]:** Port reset I/O pin y (y = 2 to 0)

13.5.12 GPIO port A secure configuration register (GPIOA_SECCFGR)

Address offset: 0x030

Reset value: 0x0000 FFFF

When the system is secure (TZEN = 1), this register provides write access security and can be written only by a secure access operation. It is used to configure a selected I/O as secure. A nonsecure write access operation to this register is discarded.

When the system is not secure (TZEN = 0), this register is RAZ/WI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	Res.	Res.	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:5 **SEC[15:5]**: I/O pin of port secure bit enable y (y = 15 to 5)
These bits are written by software to enabled the security I/O port pin.
0: The I/O pin is nonsecure
1: The I/O pin is secure. Refer to *Table 120* for all corresponding secured bits.

Bits 4:3 Reserved, must be kept at reset value.

Bits 2:0 **SEC[2:0]**: I/O pin of port secure bit enable y (y = 2 to 0)
These bits are written by software to enabled the security I/O port pin.
0: The I/O pin is nonsecure
1: The I/O pin is secure. Refer to *Table 120* for all corresponding secured bits.

13.6 GPIO port B registers

This section gives a detailed description of the GPIO port B registers.

The peripheral registers can be written in word, half-word, or byte mode.

13.6.1 GPIO port B mode register (GPIOB_MODER)

Address offset: 0x000

Reset value: 0xC30F 3EB0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		Res.	Res.	Res.	Res.	MODE12[1:0]		Res.	Res.	Res.	Res.	MODE9[1:0]		MODE8[1:0]	
rw	rw					rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		Res.	Res.	Res.	Res.
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:30 **MODE15[1:0]:** Port configuration I/O pin 15
These bits are written by software to configure the I/O mode.
Access can be protected by GPIOB SEC15.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **MODE12[1:0]:** Port configuration I/O pin 12

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:18 **MODE9[1:0]:** Port configuration I/O pin 9

Bits 17:16 **MODE8[1:0]:** Port configuration I/O pin 8

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **MODE6[1:0]:** Port configuration I/O pin 6

Bits 11:10 **MODE5[1:0]:** Port configuration I/O pin 5

Bits 9:8 **MODE4[1:0]:** Port configuration I/O pin 4

Bits 7:6 **MODE3[1:0]:** Port configuration I/O pin 3

Bits 5:4 **MODE2[1:0]:** Port configuration I/O pin 2

Bits 3:0 Reserved, must be kept at reset value.

13.6.2 GPIO port B output type register (GPIOB_OTYPER)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	Res.	Res.	OT12	Res.	Res.	OT9	OT8	Res.	OT6	OT5	OT4	OT3	OT2	Res.	Res.
rw			rw			rw	rw		rw	rw	rw	rw	rw		

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OT[15]:** Port configuration I/O pin y (y = 15)
These bits are written by software to configure the I/O output type.
Access can be protected by GPIOB SEC15.
0: Output push-pull (reset state)
1: Output open-drain

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **OT[12]:** Port configuration I/O pin y (y = 12)

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **OT[9:8]:** Port configuration I/O pin y (y = 9 to 8)

Bit 7 Reserved, must be kept at reset value.

Bits 6:2 **OT[6:2]:** Port configuration I/O pin y (y = 6 to 2)

Bits 1:0 Reserved, must be kept at reset value.

13.6.3 GPIO port B output speed register (GPIOB_OSPEEDR)

Address offset: 0x008

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15[1:0]		Res.	Res.	Res.	Res.	OSPEED12[1:0]		Res.	Res.	Res.	Res.	OSPEED9[1:0]		OSPEED8[1:0]	
rw	rw					rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		Res.	Res.	Res.	Res.
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:30 **OSPEED15[1:0]**: Port configuration I/O pin 15

These bits are written by software to configure the I/O output speed.
Access can be protected by GPIOB SEC15.
00: Low speed
01: Medium speed
10: High speed
11: Reserved

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **OSPEED12[1:0]:** Port configuration I/O pin 12

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:18 **OSPEED9[1:0]:** Port configuration I/O pin 9

Bits 17:16 **OSPEED8[1:0]:** Port configuration I/O pin 8

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **OSPEED6[1:0]:** Port configuration I/O pin 6

Bits 11:10 **OSPEED5[1:0]:** Port configuration I/O pin 5

Bits 9:8 **OSPEED4[1:0]:** Port configuration I/O pin 4

Bits 7:6 **OSPEED3[1:0]:** Port configuration I/O pin 3

Bits 5:4 **OSPEED2[1:0]:** Port configuration I/O pin 2

Bits 3:0 Reserved, must be kept at reset value.

13.6.4 GPIO port B pull-up/pull-down register (GPIOB_PUPDR)

Address offset: 0x00C

Reset value: 0x0000 0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		Res.	Res.	Res.	Res.	PUPD12[1:0]		Res.	Res.	Res.	Res.	PUPD9[1:0]		PUPD8[1:0]	
rw	rw					rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		Res.	Res.	Res.	Res.
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:30 **PUPD15[1:0]:** Port configuration I/O pin 15
These bits are written by software to configure the I/O pull-up or pull-down.
Access can be protected by GPIOB SEC15.
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **PUPD12[1:0]:** Port configuration I/O pin 12

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:18 **PUPD9[1:0]:** Port configuration I/O pin 9

Bits 17:16 **PUPD8[1:0]:** Port configuration I/O pin 8

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **PUPD6[1:0]:** Port configuration I/O pin 6

Bits 11:10 **PUPD5[1:0]:** Port configuration I/O pin 5

Bits 9:8 **PUPD4[1:0]:** Port configuration I/O pin 4

Bits 7:6 **PUPD3[1:0]:** Port configuration I/O pin 3

Bits 5:4 **PUPD2[1:0]:** Port configuration I/O pin 2

Bits 3:0 Reserved, must be kept at reset value.

13.6.5 GPIO port B input data register (GPIOB_IDR)

Address offset: 0x010

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	Res.	Res.	ID12	Res.	Res.	ID9	ID8	Res.	ID6	ID5	ID4	ID3	ID2	Res.	Res.
r			r			r	r		r	r	r	r	r		

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **ID[15]:** Port input data I/O pin y (y = 15)
These bits are read-only. They contain the input value of the corresponding I/O port.
Access can be protected by GPIOB SECy.

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **ID[12]:** Port input data I/O pin y (y = 12)

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **ID[9:8]:** Port input data I/O pin y (y = 9 to 8)

Bit 7 Reserved, must be kept at reset value.

Bits 6:2 **ID[6:2]:** Port input data I/O pin y (y = 6 to 2)

Bits 1:0 Reserved, must be kept at reset value.

13.6.6 GPIO port B output data register (GPIOB_ODR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	Res.	Res.	OD12	Res.	Res.	OD9	OD8	Res.	OD6	OD5	OD4	OD3	OD2	Res.	Res.
rw			rw			rw	rw		rw	rw	rw	rw	rw		

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OD[15]:** Port output data I/O pin y (y = 15)
These bits can be read and written by software.
Access can be protected by GPIOB SECy.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOB_BSRR or GPIOB_BRR registers.

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **OD[12]:** Port output data I/O pin y (y = 12)

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **OD[9:8]:** Port output data I/O pin y (y = 9 to 8)

Bit 7 Reserved, must be kept at reset value.

Bits 6:2 **OD[6:2]:** Port output data I/O pin y (y = 6 to 2)

Bits 1:0 Reserved, must be kept at reset value.

13.6.7 GPIO port B bit set/reset register (GPIOB_BSRR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	Res.	Res.	BR12	Res.	Res.	BR9	BR8	Res.	BR6	BR5	BR4	BR3	BR2	Res.	Res.
w			w			w	w		w	w	w	w	w		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	Res.	Res.	BS12	Res.	Res.	BS9	BS8	Res.	BS6	BS5	BS4	BS3	BS2	Res.	Res.
w			w			w	w		w	w	w	w	w		

Bit 31 **BR[15]:** Port reset I/O pin y (y = 15)
These bits are write-only. A read to these bits returns the value 0.
Access can be protected by GPIOB SECy.
0: No action on the corresponding ODy bit
1: Resets the corresponding ODy bit
If both BSy and BRy are set, BSy has priority.

Bits 30:29 Reserved, must be kept at reset value.

Bit 28 **BR[12]:** Port reset I/O pin y (y = 12)

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **BR[9:8]:** Port reset I/O pin y (y = 9 to 8)

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **BR[6:2]:** Port reset I/O pin y (y = 6 to 2)

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **BS[15]:** Port set I/O pin y (y = 15)
These bits are write-only. A read to these bits returns the value 0.
Access can be protected by GPIOB SECy.
0: No action on the corresponding ODy bit
1: Sets the corresponding ODy bit

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **BS[12]:** Port set I/O pin y (y = 12)

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **BS[9:8]:** Port set I/O pin y (y = 9 to 8)

Bit 7 Reserved, must be kept at reset value.

Bits 6:2 **BS[6:2]:** Port set I/O pin y (y = 6 to 2)

Bits 1:0 Reserved, must be kept at reset value.

13.6.8 GPIO port B configuration lock register (GPIOB_LCKR)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the

GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOB_LCKR register. Only (32-bit) word access is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	Res.	Res.	LCK12	Res.	Res.	LCK9	LCK8	Res.	LCK6	LCK5	LCK4	LCK3	LCK2	Res.	Res.
rw			rw			rw	rw		rw	rw	rw	rw	rw		

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.
Access can be protected by any GPIOB SECy.
0: Port configuration lock key not active
1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.
- LOCK key write sequence:
WR LCKR[16] = 1 + LCKR[15:0]
WR LCKR[16] = 0 + LCKR[15:0]
WR LCKR[16] = 1 + LCKR[15:0]
- LOCK key read
RD LCKR[16] = 1 (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCKR[15:0] must not change.
Any error in the lock sequence aborts the LOCK.
After the first LOCK sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bit 15 **LCK[15]:** Port lock I/O pin y (y = 15)

These bits are read/write but can only be written when the LCKK bit is 0.
Access can be protected by GPIOB SECy.
0: Port configuration not locked
1: Port configuration locked

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **LCK[12]:** Port lock I/O pin y (y = 12)

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **LCK[9:8]:** Port lock I/O pin y (y = 9 to 8)

Bit 7 Reserved, must be kept at reset value.

Bits 6:2 **LCK[6:2]:** Port lock I/O pin y (y = 6 to 2)

Bits 1:0 Reserved, must be kept at reset value.

13.6.9 GPIO port B alternate function low register (GPIOB_AFRL)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **AFSEL6[3:0]:** Alternate function selection for port I/O pin 6
These bits are written by software to configure alternate function I/Os.
Access can be protected by GPIOB SEC6.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

Bits 23:20 **AFSEL5[3:0]:** Alternate function selection for port I/O pin 5

Bits 19:16 **AFSEL4[3:0]:** Alternate function selection for port I/O pin 4

Bits 15:12 **AFSEL3[3:0]:** Alternate function selection for port I/O pin 3

Bits 11:8 **AFSEL2[3:0]:** Alternate function selection for port I/O pin 2

Bits 7:0 Reserved, must be kept at reset value.

13.6.10 GPIO port B alternate function high register (GPIOB_AFRH)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFSEL12[3:0]			
rw	rw	rw	rw									rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFSEL9[3:0]				AFSEL8[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **AFSEL15[3:0]:** Alternate function selection for port I/O pin 15
These bits are written by software to configure alternate function I/Os.
Access can be protected by GPIOB SEC15.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

Bits 27:20 Reserved, must be kept at reset value.

Bits 19:16 **AFSEL12[3:0]:** Alternate function selection for port I/O pin 12

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:4 **AFSEL9[3:0]:** Alternate function selection for port I/O pin 9

Bits 3:0 **AFSEL8[3:0]:** Alternate function selection for port I/O pin 8

13.6.11 GPIO port B bit reset register (GPIOB_BRR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	Res.	Res.	BR12	Res.	Res.	BR9	BR8	Res.	BR6	BR5	BR4	BR3	BR2	Res.	Res.
w			w			w	w		w	w	w	w	w		

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **BR[15]:** Port reset I/O pin y (y = 15)
These bits are write-only. A read to these bits returns the value 0.
Access can be protected by GPIOB SECy.
0: No action on the corresponding ODy bit
1: Reset the corresponding ODy bit

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **BR[12]:** Port reset I/O pin y (y = 12)

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **BR[9:8]:** Port reset I/O pin y (y = 9 to 8)

Bit 7 Reserved, must be kept at reset value.

Bits 6:2 **BR[6:2]:** Port reset I/O pin y (y = 6 to 2)

Bits 1:0 Reserved, must be kept at reset value.

13.6.12 GPIO port B secure configuration register (GPIOB_SECCFGR)

Address offset: 0x030

Reset value: 0x0000 FFFF

When the system is secure (TZEN = 1), this register provides write access security and can be written only by secure access. It is used to configure a selected I/O as secure. A nonsecure write access operation to this register is discarded.

When the system is not secure (TZEN = 0), this register is RAZ/WI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	Res.	Res.	SEC12	Res.	Res.	SEC9	SEC8	Res.	SEC6	SEC5	SEC4	SEC3	SEC2	Res.	Res.
rw			rw			rw	rw		rw	rw	rw	rw	rw		

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **SEC[15]**: I/O pin of port secure bit enable y (y = 15)
These bits are written by software to enabled the security I/O port pin.
0: The I/O pin is nonsecure
1: The I/O pin is secure. Refer to *Table 120* for all corresponding secured bits.

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **SEC[12]**: I/O pin of port secure bit enable y (y = 12)
These bits are written by software to enabled the security I/O port pin.
0: The I/O pin is nonsecure
1: The I/O pin is secure. Refer to *Table 120* for all corresponding secured bits.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **SEC[9:8]**: I/O pin of port secure bit enable y (y = 9 to 8)
These bits are written by software to enabled the security I/O port pin.
0: The I/O pin is nonsecure
1: The I/O pin is secure. Refer to *Table 120* for all corresponding secured bits.

Bit 7 Reserved, must be kept at reset value.

Bits 6:2 **SEC[6:2]**: I/O pin of port secure bit enable y (y = 6 to 2)
These bits are written by software to enabled the security I/O port pin.
0: The I/O pin is nonsecure
1: The I/O pin is secure. Refer to *Table 120* for all corresponding secured bits.

Bits 1:0 Reserved, must be kept at reset value.

13.7 GPIO port C registers

This section gives a detailed description of the GPIO port C registers.

The peripheral registers can be written in word, half-word, or byte mode.

13.7.1 GPIO port C mode register (GPIOC_MODER)

Address offset: 0x000

Reset value: 0xFC00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 **MODE15[1:0]:** Port C configuration I/O pin 15
These bits are written by software to configure the I/O mode.
Access can be protected by GPIOC SEC15.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

Bits 29:28 **MODE14[1:0]:** Port C configuration I/O pin 14

Bits 27:26 **MODE13[1:0]:** Port C configuration I/O pin 13

Bits 25:0 Reserved, must be kept at reset value.

13.7.2 GPIO port C output type register (GPIOC_OTYPER)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **OT[15:13]:** Port C configuration I/O pin y (y = 15 to 13)
These bits are written by software to configure the I/O output type.
Access can be protected by GPIOC SECy.
0: Output push-pull (reset state)
1: Output open-drain

Bits 12:0 Reserved, must be kept at reset value.

13.7.3 GPIOC port output speed register (GPIOC_OSPEEDR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	OSPEED13[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:28 Reserved, must be kept at reset value.(1)

Bits 27:26 **OSPEED13[1:0]:** Port C configuration I/O pin 13
These bits are written by software to configure the I/O output speed.
Access can be protected by GPIOC SEC15.
00: Low speed
01: Medium speed
10: High speed
11: Reserved

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

Bits 25:0 Reserved, must be kept at reset value.

1. PC15 and PC14 OSPEED are fixed to Low speed.

13.7.4 GPIO port C pull-up/pull-down register (GPIOC_PUPDR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 **PUPD15[1:0]:** Port C configuration I/O pin 15
These bits are written by software to configure the I/O pull-up or pull-down.
Access can be protected by GPIOC SEC15.
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

Bits 29:28 **PUPD14[1:0]:** Port C configuration I/O pin 14

Bits 27:26 **PUPD13[1:0]:** Port C configuration I/O pin 13

Bits 25:0 Reserved, must be kept at reset value.

13.7.5 GPIO port C input data register (GPIOC_IDR)

Address offset: 0x010

Reset value: 0x0000 X000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r													

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **ID[15:13]:** Port C input data I/O pin y (y = 15 to 13)
These bits are read-only. They contain the input value of the corresponding I/O port.
Access can be protected by GPIOC SECy.

Bits 12:0 Reserved, must be kept at reset value.

13.7.6 GPIO port C output data register (GPIOC_ODR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **OD[15:13]:** Port C output data I/O pin y (y = 15 to 13)
These bits can be read and written by software.
Access can be protected by GPIOC SECy.
Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOC_BSRR or GPIOC_BRR registers.

Bits 12:0 Reserved, must be kept at reset value.

13.7.7 GPIO port C bit set/reset register (GPIOC_BSRR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w													

Bits 31:29 **BR[15:13]:** Port C reset I/O pin y (y = 15 to 13)
These bits are write-only. A read to these bits returns the value 0.
Access can be protected by GPIOC SECy.
0: No action on the corresponding ODy bit
1: Resets the corresponding ODy bit
Note: If both BSy and BRy are set, BSy has priority.

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:13 **BS[15:13]:** Port C set I/O pin y (y = 15 to 13)
These bits are write-only. A read to these bits returns the value 0.
Access can be protected by GPIOC SECy.
0: No action on the corresponding ODy bit
1: Sets the corresponding ODy bit

Bits 12:0 Reserved, must be kept at reset value.

13.7.8 GPIO port C configuration lock register (GPIOC_LCKR)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:13] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:13] must not change. When the LOCK sequence has been applied to a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOC_LCKR register. Only (32-bit) word access is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.
Access can be protected by any GPIOC SECy.
0: Port configuration lock key not active
1: Port configuration lock key active. The GPIOC_LCKR register is locked until the next MCU reset or peripheral reset.
- LOCK key write sequence:
WR LCKR[16] = 1 + LCKR[15:13]
WR LCKR[16] = 0 + LCKR[15:13]
WR LCKR[16] = 1 + LCKR[15:13]
- LOCK key read
RD LCKR[16] = 1 (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCK[15:13] must not change.
Any error in the lock sequence aborts the LOCK.
After the first LOCK sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:13 **LCK[15:13]:** Port C lock I/O pin y (y = 15 to 13)

These bits are read/write but can only be written when the LCKK bit is 0
Access can be protected by GPIOC SECy.
0: Port configuration not locked
1: Port configuration locked

Bits 12:0 Reserved, must be kept at reset value.

13.7.9 GPIO port C alternate function high register (GPIOC_AFRH)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:28 **AFSEL15[3:0]:** Alternate function selection for port C I/O pin 15
These bits are written by software to configure alternate function I/Os.
Access can be protected by GPIOC SEC15.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

Bits 27:24 **AFSEL14[3:0]:** Alternate function selection for port C I/O pin 14

Bits 23:20 **AFSEL13[3:0]:** Alternate function selection for port C I/O pin 13

Bits 19:0 Reserved, must be kept at reset value.

13.7.10 GPIO port C bit reset register (GPIOC_BRR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w													

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **BR[15:13]:** Port reset I/O pin y (y = 15 to 13)
These bits are write-only. A read to these bits returns the value 0.
Access can be protected by GPIOC SECy.
0: No action on the corresponding ODy bit
1: Reset the corresponding ODy bit

Bits 12:0 Reserved, must be kept at reset value.

13.7.11 GPIO port C secure configuration register (GPIOC_SECCFGR)

Address offset: 0x030

Reset value: 0x0000 FFFF

When the system is secure (TZEN = 1), this register provides write access security and can be written only by a secure access. It is used to configure a selected I/O as secure. A nonsecure write access operation to this register is discarded.

When the system is not secure (TZEN = 0), this register is RAZ/WI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **SEC[15:13]**: I/O pin of port C secure bit enable y (y = 15 to 13)
These bits are written by software to enabled the security I/O port pin.
0: The I/O pin is nonsecure
1: The I/O pin is secure. Refer to *Table 120* for all corresponding secured bits.

Bits 12:0 Reserved, must be kept at reset value.

13.8 GPIO port H registers

This section gives a detailed description of the GPIO port H registers.

The peripheral registers can be written in word, half-word, or byte mode.

13.8.1 GPIO port H mode register (GPIOH_MODER)

Address offset: 0x000

Reset value: 0x0000 00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
								rw	rw						

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **MODE3[1:0]:** Port H configuration I/O pin 3
These bits are written by software to configure the I/O mode.
Access can be protected by GPIOH SEC3.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

Bits 5:0 Reserved, must be kept at reset value.

13.8.2 GPIO port H output type register (GPIOH_OTYPER)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT3	Res.	Res.	Res.
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **OT3:** Port H configuration I/O pin 3
This bit is written by software to configure the I/O output type.
Access can be protected by GPIOH SEC3.
0: Output push-pull (reset state)
1: Output open-drain

Bits 2:0 Reserved, must be kept at reset value.

13.8.3 GPIO port H output speed register (GPIOH_OSPEEDR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPEED3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
								rw	rw						

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **OSPEED3[1:0]**: Port H configuration I/O pin 3

These bits are written by software to configure the I/O output speed.
Access can be protected by GPIOH SEC3.
00: Low speed
01: Medium speed
10: High speed
11: Reserved

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

Bits 5:0 Reserved, must be kept at reset value.

13.8.4 GPIO port H pull-up/pull-down register (GPIOH_PUPDR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PUPD3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
								rw	rw						

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **PUPD3[1:0]:** Port H configuration I/O pin 3

These bits are written by software to configure the I/O pull-up or pull-down.
Access can be protected by GPIOH SEC3.
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

Bits 5:0 Reserved, must be kept at reset value.

13.8.5 GPIO port H input data register (GPIOH_IDR)

Address offset: 0x010

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID3	Res.	Res.	Res.
												r			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ID3:** Port H input data I/O pin 3
This bit is read-only. It contain the input value of the corresponding I/O port.
Access can be protected by GPIOH SEC3.

Bits 2:0 Reserved, must be kept at reset value.

13.8.6 GPIO port H output data register (GPIOH_ODR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD3	Res.	Res.	Res.
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **OD3:** Port H output data I/O pin 3
This bits can be read and written by software.
Access can be protected by GPIOH SEC3.
Note: For atomic bit set/reset, the OD bit can be individually set and/or reset by writing to the GPIOH_BSRR or GPIOH_BRR registers.

Bits 2:0 Reserved, must be kept at reset value.

13.8.7 GPIO port H bit set/reset register (GPIOH_BSRR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR3	Res.	Res.	Res.
												w			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BS3	Res.	Res.	Res.
												w			

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **BR3:** Port H reset I/O pin 3
This bit is write-only. A read to this bit returns the value 0.
Access can be protected by GPIOH SEC3.
0: No action on the corresponding OD3 bit
1: Resets the corresponding OD3 bit
Note: If both BS3 and BR3 are set, BS3 has priority.

Bits 18:4 Reserved, must be kept at reset value.

Bit 3 **BS3:** Port H set I/O pin 3
This bit is write-only. A read to this bit returns the value 0.
Access can be protected by GPIOH SEC3.
0: No action on the corresponding OD3 bit
1: Sets the corresponding OD3 bit

Bits 2:0 Reserved, must be kept at reset value.

13.8.8 GPIO port H configuration lock register (GPIOH_LCKR)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bit [3] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[3] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOH_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCK3	Res.	Res.	Res.
												rw			

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key
This bit can be read any time. It can only be modified using the lock key write sequence.
Access can be protected by GPIOH SEC3.
0: Port configuration lock key not active
1: Port configuration lock key active. The GPIOH_LCKR register is locked until the next MCU reset or peripheral reset.
- LOCK key write sequence:
WR LCKR[16] = 1 + LCKR[3]
WR LCKR[16] = 0 + LCKR[3]
WR LCKR[16] = 1 + LCKR[3]
- LOCK key read
RD LCKR[16] = 1 (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCK3 must not change.
Any error in the lock sequence aborts the LOCK.
After the first LOCK sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **LCK3:** Port H lock I/O pin 3
This bit is read/write but can only be written when the LCKK bit is 0.
Access can be protected by GPIOH SEC3.
0: Port configuration not locked
1: Port configuration locked

Bits 2:0 Reserved, must be kept at reset value.

13.8.9 GPIO port H alternate function low register (GPIOH_AFRL)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw												

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **AFSEL3[3:0]:** Alternate function selection for port H I/O pin 3
These bits are written by software to configure alternate function I/Os.
Access can be protected by GPIOH SEC3.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

Bits 11:0 Reserved, must be kept at reset value.

13.8.10 GPIO port H bit reset register (GPIOH_BRR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR3	Res.	Res.	Res.
												w			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **BR3:** Port H reset I/O pin 3
This bit is write-only. A read to this bit returns the value 0.
Access can be protected by GPIOH SEC3.
0: No action on the corresponding OD3 bit
1: Reset the corresponding OD3 bit

Bits 2:0 Reserved, must be kept at reset value.

13.8.11 GPIO port H secure configuration register (GPIOH_SECCFGR)

Address offset: 0x030

Reset value: 0x0000 FFFF

When the system is secure (TZEN = 1), this register provides write access security and can be written only by a secure access. It is used to configure a selected I/O as secure. A nonsecure write access operation to this register is discarded.

When the system is not secure (TZEN = 0), this register is RAZ/WI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC3	Res.	Res.	Res.
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **SEC3**: I/O pin of port H secure bit enable 3
This bit is written by software to enabled the security I/O port pin.
0: The I/O pin is nonsecure
1: The I/O pin is secure. Refer to *Table 120* for all corresponding secured bits.

Bits 2:0 Reserved, must be kept at reset value.

13.8.12 GPIOA register map

Table 121. GPIOA register map and reset values

Offset	Register	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
0x000	**GPIOA_MODER**	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		Res.	Res.	Res.	Res.	MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1					1	1	1	1	1	1
0x004	**GPIOA_OTYPER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	Res.	Res.	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0			0	0	0
0x008	**GPIOA_OSPEEDR**	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		Res.	Res.	Res.	Res.	OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0
0x00C	**GPIOA_PUPDR**	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		Res.	Res.	Res.	Res.	PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0
0x010	**GPIOA_IDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	Res.	Res.	ID2	ID1	ID0
	Reset value																	X	X	X	X	X	X	X	X	X	X	X			X	X	X
0x014	**GPIOA_ODR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	Res.	Res.	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0			0	0	0
0x018	**GPIOA_BSRR**	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	Res.	Res.	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	Res.	Res.	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0
0x01C	**GPIOA_LCKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	Res.	Res.	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0			0	0	0
0x020	**GPIOA_AFRL**	AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0
0x024	**GPIOA_AFRH**	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]				AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**GPIOA_BRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	Res.	Res.	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0			0	0	0
0x02C	Reserved	Reserved																															
0x030	**GPIOA_SECCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	Res.	Res.	SEC2	SEC1	SEC0
	Reset value																	1	1	1	1	1	1	1	1	1	1	1			1	1	1
0x034 to 0x3FC	Reserved	Reserved																															

13.8.13 GPIOB register map

Table 122. GPIOB register map and reset values

Offset	Register	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
0x000	**GPIOB_MODER**	MODE15[1:0]		Res.	Res.	Res.	Res.	MODE12[1:0]		Res.	Res.	Res.	Res.	MODE9[1:0]		MODE8[1:0]		Res.	Res.	MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		Res.	Res.	Res.	Res.
	Reset value	1	1					1	1					1	1	1	1			1	1	1	1	1	0	1	0	1	1				
0x004	**GPIOB_OTYPER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	Res.	Res.	OT12	Res.	Res.	OT9	OT8	Res.	OT6	OT5	OT4	OT3	OT2	Res.	Res.
	Reset value																	0			0			0	0		0	0	0	0	0		
0x008	**GPIOB_OSPEEDR**	OSPEED15[1:0]		Res.	Res.	Res.	Res.	OSPEED12[1:0]		Res.	Res.	Res.	Res.	OSPEED9[1:0]		OSPEED8[1:0]		Res.	Res.	OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		Res.	Res.	Res.	Res.
	Reset value	0	0					0	0					0	0	0	0			0	0	0	0	0	0	1	0	0	0				
0x00C	**GPIOB_PUPDR**	PUPD15[1:0]		Res.	Res.	Res.	Res.	PUPD12[1:0]		Res.	Res.	Res.	Res.	PUPD9[1:0]		PUPD8[1:0]		Res.	Res.	PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		Res.	Res.	Res.	Res.
	Reset value	0	0					0	0					0	0	0	0			0	0	0	0	0	1	0	0	0	0				
0x010	**GPIOB_IDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	Res.	Res.	ID12	Res.	Res.	ID9	ID8	Res.	ID6	ID5	ID4	ID3	ID2	Res.	Res.
	Reset value																	X			X			X	X		X	X	X	X	X		
0x014	**GPIOB_ODR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	Res.	Res.	OD12	Res.	Res.	OD9	OD8	Res.	OD6	OD5	OD4	OD3	OD2	Res.	Res.
	Reset value																	0			0			0	0		0	0	0	0	0		
0x018	**GPIOB_BSRR**	BR15		Res.	BR12	Res.	Res.	BR9	BR8	Res.	BR6	BR5	BR4	BR3	BR2	Res.	Res.	BS15	Res.	Res.	BS12	Res.	Res.	BS9	BS8	Res.	BS6	BS5	BS4	BS3	BS2	Res.	Res.
	Reset value	0			0			0	0		0	0	0	0	0			0			0			0	0		0	0	0	0	0		
0x01C	**GPIOB_LCKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	Res.	Res.	LCK12	Res.	Res.	LCK9	LCK8	Res.	LCK6	LCK5	LCK4	LCK3	LCK2	Res.	Res.
	Reset value																0	0			0			0	0		0	0	0	0	0		
0x020	**GPIOB_AFRL**	Res.	Res.	Res.	Res.	AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]				AFSEL3[3:0]				AFSEL2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x024	**GPIOB_AFRH**	AFSEL15[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFSEL12[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFSEL9[3:0]				AFSEL8[3:0]			
	Reset value	0	0	0	0									0	0	0	0									0	0	0	0	0	0	0	0
0x028	**GPIOB_BRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	Res.	Res.	BR12	Res.	Res.	BR9	BR8	Res.	BR6	BR5	BR4	BR3	BR2	Res.	Res.
	Reset value																	0			0			0	0		0	0	0	0	0		
0x02C	Reserved	Reserved																															
0x030	**GPIOB_SECCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC15	Res.	Res.	SEC12	Res.	Res.	SEC9	SEC8	Res.	SEC6	SEC5	SEC4	SEC3	SEC2	Res.	Res.
	Reset value																	1			1			1	1		1	1	1	1	1		
0x034 to 0x3FC	Reserved	Reserved																															

13.8.14 GPIOC register map

Table 123. GPIOC register map and reset values

Offset	Register	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
0x000	**GPIOC_MODER**	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	1	1	1	1	1	1																										
0x004	**GPIOC_OTYPER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0	0													
0x008	**GPIOC_OSPEEDR**	Res.	Res.	Res.	Res.	OSPEED13[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0																										
0x00C	**GPIOC_PUPDR**	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0																										
0x010	**GPIOC_IDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	X	X	X													
0x014	**GPIOC_ODR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0	0													
0x018	**GPIOC_BSRR**	BR15	BR14	BR13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BS15	BS14	BS13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0														0	0	0													
0x01C	**GPIOC_LCKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																0	0	0	0													
0x020	Reserved	Reserved																															
0x024	**GPIOC_AFRH**	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0																				
0x028	**GPIOC_BRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0	0													
0x02C	Reserved	Reserved																															
0x030	**GPIOC_SECCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC15	SEC14	SEC13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	1	1	1													
0x034 to 0x3FC	Reserved	Reserved																															

13.8.15 GPIOH register map

Table 124. GPIOH register map and reset values

Offset	Register	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
0x000	**GPIOH_MODER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																									1	1						
0x004	**GPIOH_OTYPER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT3	Res.	Res.	Res.
	Reset value																													0			
0x008	**GPIOH_OSPEEDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPEED3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																									0	0						
0x00C	**GPIOH_PUPDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PUPD3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																									0	0						
0x010	**GPIOH_IDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID3	Res.	Res.	Res.
	Reset value																													X			
0x014	**GPIOH_ODR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD3	Res.	Res.	Res.
	Reset value																													0			
0x018	**GPIOH_BSRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BS3	Res.	Res.	Res.
	Reset value													0																0			
0x01C	**GPIOH_LCKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCK3	Res.	Res.	Res.
	Reset value																0													0			
0x020	**GPIOH_AFRL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFSEL3[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0	0	0												
0x024	Reserved	Reserved																															
0x028	**GPIOH_BRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR3	Res.	Res.	Res.
	Reset value																													0			
0x02C	Reserved	Reserved																															
0x030	**GPIOH_SECCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC3	Res.	Res.	Res.
	Reset value																													1			
0x034 to 0x3FC	Reserved	Reserved																															

Refer to *Section 2.3.2: Memory map and register boundary addresses* for the register boundary addresses.

14 System configuration controller (SYSCFG)

14.1 SYSCFG main features

The devices feature a set of configuration registers. The main purposes of the system configuration controller are the following:

- Managing robustness feature
- Configuring FPU interrupts
- Enabling/disabling the I^2C fast mode plus high-drive mode on some I/Os and voltage booster for I/Os analog switches
- Managing the I/O compensation cell
- Provides memory erase status
- 2.4 GHz RADIO external PA bypass signal configuration
- Flash abort operation interrupt enable
- TrustZone® security support

14.2 SYSCFG functional description

14.2.1 I/O compensation cell management

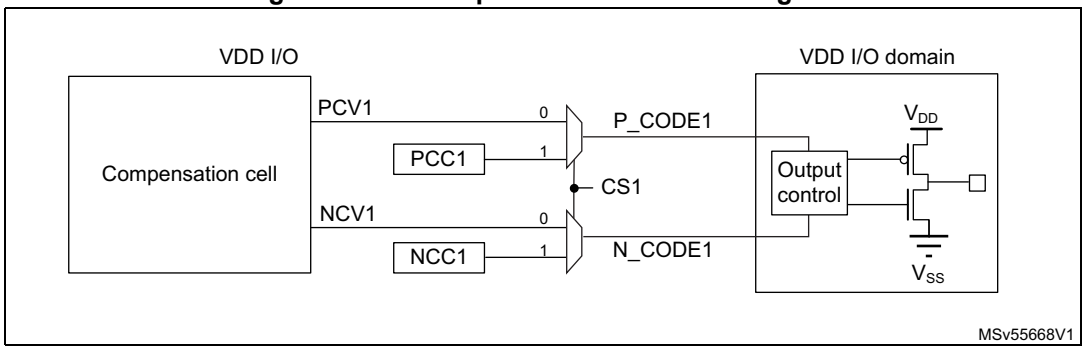
The I/O compensation cell generates an 8-bit value for the I/O buffer (4 bits for N-MOS and 4 bits for P-MOS), which depends on PVT operating conditions such as process, voltage, and temperature. These bits are used to control the current slew-rate and output impedance in the I/O buffer.

By default the compensation cell is disabled, and a fixed code is applied to all the I/Os.

When enabled, the compensation cell tracks the PVT, and the 8-bit code PCV1 and NCV1 for I/Os supplied by V_{DD} is available in SYSCFG_CCVR once the RDY1 is set. If the CS1 bit is cleared, the I/O receives the code from SYSCFG_CCVR, resulting from the compensation cell.

To optimize the trimming, the code can be adjusted through SYSCFG_CCCR. A set of bits is available, namely PCC1/NCC1 for the V_{DD} power rail. It can be selected through the CS1 bit in SYSCFG_CCCSR (see *Figure 47*).

Figure 47. I/O compensation cell block diagram



To reduce the power consumption, it is recommended to copy the code from SYSCFG_CCVR to SYSCFG_CCCR. After the result is ready, set the CS bit and disable the compensation cell.

Table 125. Effect of low-power modes on I/O compensation

Mode	Description
Sleep	Compensation value applied on I/Os. Compensation cell controlled by the EN bit of the SYSCFG_CCCSR register.
Stop	
Standby	Default compensation value applied on I/Os. Compensation cell disabled.

14.2.2 SYSCFG TrustZone® security and privilege

SYSCFG TrustZone security

When the TrustZone security is activated, the SYSCFG is able to secure registers from being modified by nonsecure access.

The TrustZone security is activated by the TZEN user option bit in the FLASH_OPTR register.

Nonsecure read/write access to a secured register is RAZ/WI and generates an illegal access event. An illegal access interrupt is generated if the SYSCFG illegal access event is enabled in the GTZC1_TZIC.

As soon as at least one function is configured to be secured, the SYSCFG reset and clock control bits in the RCC are also secured.

Privileged/unprivileged mode

The SYSCFG registers can be read and written by privileged or unprivileged access except the SYSCFG registers for CPU configuration: SYSCFG_CSLCKR, SYSCFG_FPUIMR and SYSCFG_CNSLCKR registers, and the FPUSEC bit in the SYSCFG_SECCFGR register.

An unprivileged access to a privileged register is RAZ/WI.

The table below shows the register security overview.

Table 126. TrustZone security and privilege register access

SYSCFG register name	Read/write access		Privileged /unprivileged access
TrustZone configuration(1)	TZEN = 1	TZEN = 0	Not applicable
SYSCFG_SECCFGR	Read: no restriction Write: secure access only Nonsecure write is WI and generates an illegal access event.	RAZ/WI	Read: no restriction FPUSEC privileged write only Other bits write: no restriction
SYSCFG_CSLCKR	Read/Write: secure access only Nonsecure access is RAZ/WI and generates an illegal access event.	RAZ/WI	Privileged only Unprivileged: RAZ/WI

Table 126. TrustZone security and privilege register access (continued)

SYSCFG register name	Read/write access		Privileged /unprivileged access
TrustZone configuration(1)	TZEN = 1	TZEN = 0	Not applicable
SYSCFG_FPUIMR	– If FPUSEC = 1: Read/Write: secure access only Nonsecure access is RAZ/WI and generates an illegal access event. – If FPUSEC = 0: Read/Write, no restriction	No restriction	Privileged only Unprivileged: RAZ/WI
SYSCFG_CNSLCKR	Read/write: no restriction	No restriction	Privileged only Unprivileged: RAZ/WI
SYSCFG_CFGR1	Read/Write: secure access only for secure bits depending on peripheral security bits in the GTZC1_TZSC Nonsecure access only for nonsecure bits, otherwise RAZ/WI	No restriction	No restriction
SYSCFG_CFGR2	– If CLASSBSEC = 1: Read/Write: secure access only Nonsecure access is RAZ/WI and generates an illegal access event. – If CLASSBSEC = 0: Read/Write, no restriction	No restriction	No restriction
SYSCFG_MESR	– If SYSCFGSEC = 1: Read/Write: secure access only Nonsecure access is RAZ/WI and generates an illegal access event. – If SYSCFGSEC = 0: Read/Write, no restriction	No restriction	No restriction
SYSCFG_CCCSR SYSCFG_CCVR SYSCFG_CCCR	– If SYSCFGSEC = 1: Read/Write: secure access only Nonsecure access is RAZ/WI and generates an illegal access event – If SYSCFGSEC = 0:Read/Write, no restriction	No restriction	No restriction

1. TrustZone security is activated by the TZEN user option bit in the FLASH_OPTR register.

14.3 SYSCFG registers

14.3.1 SYSCFG secure configuration register (SYSCFG_SECCFGR)

Address offset: 0x000

Reset value: 0x0000 0000

When the system is secure (TZEN = 1), this register provides write access security and can be written only when the access is secure. It can be globally write-protected, or each bit of this register can be individually write-protected. A nonsecure write access is WI and generates an illegal access event. There are no read restrictions.

When the system is not secure (TZEN = 0), this register is RAZ/WI.

This register can be read and written by privileged and unprivileged access, except for FPUSEC that can be written only with privileged access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPUSEC	Res.	CLASSBSEC	SYSCFGSEC
												rw		rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FPUSEC**: FPU security
0: SYSCFG_FPUIMR register can be read and written by secure and nonsecure access.
1: SYSCFG_FPUIMR register can be read and written by secure access only.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CLASSBSEC**: Class B security
0: SYSCFG_CFGR2 register can be read and written by secure and nonsecure access.
1: SYSCFG_CFGR2 register can be read and written by secure access only.

Bit 0 **SYSCFGSEC**: SYSCFG clock control, memory erase status and compensation cell registers security
0: SYSCFG configuration clock in RCC registers, SYSCFG_MESR and SYSCFG_CCCSR, SYSCFG_CCVR and SYSCFG_CCCR can be read and written by secure and nonsecure access.
1: SYSCFG configuration clock in RCC registers, SYSCFG_MESR and SYSCFG_CCCSR, SYSCFG_CCVR and SYSCFG_CCCR can be read and written by secure access only.

14.3.2 SYSCFG configuration register 1 (SYSCFG_CFGR1)

Address offset: 0x004

Reset value: 0x0000 0000

When the system is secure (TZEN = 1), this register can be a mix of secure and nonsecure bits depending on the security configuration bit in the GTZC1_TZSC peripheral and the

GPIO port pin security configuration in the GPIO peripheral. A nonsecure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN = 0), there is no access restriction.

This register can be read and written by privileged and unprivileged access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RF_EXTPABYP2_INV	RF_EXTPABYP1_INV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PB3_FMP	PA15_FMP	PA7_FMP	PA6_FMP
rw	rw											rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	ANASWVDD	BOOSTEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw	rw								

Bit 31 **RF_EXTPABYP2_INV**: signal RF_EXTPABYP2 inversion selection.

This bit can be read and written only with secure access if GTZC1_TZSC RADIOSEC is secure. This bit selects the inversion or not of the RF_EXTPABYP2 signal.

0: RF_EXTPABYP2 not inverted.

1: RF_EXTPABYP2 inverted.

Bit 30 **RF_EXTPABYP1_INV**: signal RF_EXTPABYP1 inversion selection.

This bit can be read and written only with secure access if GTZC1_TZSC RADIOSEC is secure. This bit selects the inversion or not of the RF_EXTPABYP1 signal.

0: RF_EXTPABYP1 not inverted.

1: RF_EXTPABYP1 inverted.

Bits 29:20 Reserved, must be kept at reset value.

Bit 19 **PB3_FMP**: Fast mode plus drive capability activation on PB3

This bit can be read and written only with secure access if PB3 is secure in GPIOB. This bit enables the fast mode plus drive mode for PB3 when PB3 is not used by I2C peripheral. This can be used to dive a LED for instance.

0: PB3 pin operates in standard mode when not used by I2C peripheral

1: Fast mode plus mode is enabled on PAB3 pin and the GPIO speed control is bypassed.

Bit 18 **PA15_FMP**: Fast mode plus drive capability activation on PA15

This bit can be read and written only with secure access if PA15 is secure in GPIOA. This bit enables the fast mode plus drive mode for PA15 when PA15 is not used by I2C peripheral. This can be used to dive a LED for instance.

0: PA15 pin operates in standard mode when not used by I2C peripheral

1: fFst mode plus mode is enabled on PA15 pin and the GPIO speed control is bypassed.

Bit 17 **PA7_FMP**: Fast mode plus drive capability activation on PA7

This bit can be read and written only with secure access if PA7 is secure in GPIOA. This bit enables the fast mode plus drive mode for PA7 when PA7 is not used by I2C peripheral. This can be used to dive a LED for instance.

0: PA7 pin operates in standard mode when not used by I2C peripheral

1: Fast mode plus mode is enabled on PA7 pin and the GPIO speed control is bypassed.

Bit 16 **PA6_FMP**: Fast mode plus drive capability activation on PA6
This bit can be read and written only with secure access if PA6 is secure in GPIOA. This bit enables the fast mode plus drive mode for PA6 when PA6 is not used by I2C peripheral. This can be used to dive a LED for instance.
0: PA6 pin operates in standard mode when not used by I2C peripheral
1: Fast mode plus mode is enabled on PA6 pin and the GPIO speed control is bypassed.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **ANASWVDD**: GPIO analog switch control voltage selection
Access can be protected by GTZC1_TZSC ADC4SEC.
0: I/O analog switches are supplied by V_{DDA} or booster when booster is ON.
1: I/O analog switches are supplied by V_{DD}.
Note: Refer to Table 127 for settings.

Bit 8 **BOOSTEN**: I/O analog switch voltage booster enable
Access can be protected by GTZC1_TZSC ADC4SEC.
0: I/O analog switches are supplied by V_{DDA} voltage.
1: I/O analog switches are supplied by a dedicated voltage booster (supplied by V_{DD}).
Note: Refer to Table 127 for settings.

Bits 7:0 Reserved, must be kept at reset value.

Table 127 describes when the BOOSTEN and the ANASWVDD must be set or cleared, depending on the voltage settings.

Table 127. BOOSTEN and ANASWVDD set/cleared

V_{DD}	V_{DDA}	BOOSTEN	ANASWVDD
-	> 2.4 V	0	0
> 2.4 V	≤ 2.4 V		1
≤ 2.4 V		1	0

14.3.3 SYSCFG FPU interrupt mask register (SYSCFG_FPUIMR)

Address offset: 0x008

Reset value: 0x0000 001F

This register can be read and written by privileged access only. Unprivileged access is RAZ/WI.

When the system is secure (TZEN = 1), this register can be protected against nonsecure access by setting the FPUSEC bit in the SYSCFG_SECCFGR register: nonsecure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN = 0), there is no access restriction.

This register can be read and written by privileged access only. Unprivileged access is RAZ/WI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPU_IE[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **FPU_IE[5:0]**: Floating point unit interrupts enable bits
FPU_IE[5]: Inexact interrupt enable (interrupt disable at reset)
FPU_IE[4]: Input abnormal interrupt enable
FPU_IE[3]: Overflow interrupt enable
FPU_IE[2]: Underflow interrupt enable
FPU_IE[1]: Divide-by-zero interrupt enable
FPU_IE[0]: Invalid operation Interrupt enable

14.3.4 SYSCFG CPU nonsecure lock register (SYSCFG_CNSLCKR)

Address offset: 0x00C

Reset value: 0x0000 0000

This register is used to lock the configuration of MPU and VTOR registers. This register can be read and written by privileged access only. Unprivileged access is RAZ/WI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKNSMPU	LOCKNSVTOR
														rs	rs

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **LOCKNSMPU**: MPU registers lock

This bit is set by software and cleared only by a system reset. When set, this bit disables write access to nonsecure MPU_CTRL, MPU_RNR and MPU_RBAR registers.

0: Nonsecure MPU registers write enabled

1: Nonsecure MPU registers write disabled

Bit 0 **LOCKNSVTOR**: VTOR register lock

This bit is set by software and cleared only by a system reset.

0: VTOR register write enabled

1: VTOR register write disabled

14.3.5 SYSCFG CPU secure lock register (SYSCFG_CSLOCKR)

Address offset: 0x010

Reset value: 0x0000 0000

This register is used to lock the configuration of PRIS and BFHFNMINS bits in the AIRCR register, SAU, secure MPU and VTOR_S registers.

When the system is secure (TZEN = 1), this register can be written only when the access is secure. Nonsecure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN = 0), this register is RAZ/WI

This register can be read and written by privileged access only. Unprivileged access is RAZ/WI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKSAU	LOCKSMPU	LOCKSVTAIRCR
													rs	rs	rs

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKSAU**: SAU registers lock

This bit is set by software and cleared only by a system reset. When set, this bit disables write access to SAU_CTRL, SAU_RNR, SAU_RBAR and SAU_RLAR registers.

0: SAU registers write enabled

1: SAU registers write disabled

Bit 1 **LOCKSMPU**: Secure MPU registers lock

This bit is set by software and cleared only by a system reset. When set, this bit disables write access to secure MPU_CTRL, MPU_RNR and MPU_RBAR registers.

0: Secure MPU registers writes enabled

1: Secure MPU registers writes disabled

Bit 0 **LOCKSVTAIRCR**: VTOR_S register and AIRCR register bits lock

This bit is set by software and cleared only by a system reset. When set, this bit disables write access to the VTOR_S register, and the PRIS and the BFHFNMINS bits in the AIRCR register.

0: VTOR_S register PRIS and BFHFNMINS bits in the AIRCR register write enabled

1: VTOR_S register PRIS and BFHFNMINS bits in the AIRCR register write disabled

14.3.6 SYSCFG configuration register 2 (SYSCFG_CFGR2)

Address offset: 0x014

Reset value: 0x0000 0000

When the system is secure (TZEN = 1), this register can be protected against nonsecure access by setting the CLASSBSEC bit in the SYSCFG_SECCFGR register. When the CLASSBSEC bit is set, only secure access is allowed: nonsecure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN = 0), there is no access restriction.

This register can be read and written by privileged and unprivileged access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCL	PVDL	SPL	CLL
												rs	rs	rs	rs

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ECCL**: ECC lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the flash ECC double error signal connection to TIM16 and TIM17 break input.

0: ECC double error disconnected from TIM16 and TIM17 break input

1: ECC double error connected to TIM16 and TIM17 break input

Bit 2 **PVDL**: PVD lock enable bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the PVD connection to TIM16 and TIM17 break input, as well as the PVDE and PVDLS[2:0] in the PWR register.

0: PVD interrupt disconnected from TIM16 and TIM17 break input. PVDE and PVDLS[2:0] bits can be programmed by the application.

1: PVD interrupt connected to TIM16 and TIM17 break input. PVDE and PVDLS[2:0] bits are read only.

Bit 1 **SPL**: SRAM2 parity lock bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the SRAM2 parity error signal connection to TIM16 and TIM17 break inputs.

0: SRAM2 parity error disconnected from TIM16 and TIM17 break inputs

1: SRAM2 parity error connected to TIM16 and TIM17 break inputs

Bit 0 **CLL**: Cortex-M33 LOCKUP (hardfault) output enable

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the connection of Cortex-M33 LOCKUP (hardfault) output to TIM16 and TIM17 break input.

0: Cortex-M33 LOCKUP output disconnected from TIM16 to TIM17 break inputs

1: Cortex-M33 LOCKUP output connected to TIM16 and TIM17 break inputs

14.3.7 SYSCFG memory erase status register (SYSCFG_MESR)

Address offset: 0x018

Power-on reset value: 0x0000 0000

System reset value: 0x0000 000X (bit MCLKR not affected by system reset)

When the system is secure (TZEN = 1), this register can be protected against nonsecure access by setting the SYSCFGSEC bit in the SYSCFG_SECCFGR register. When the SYSCFGSEC bit is set, only secure access is allowed: nonsecure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN = 0), there is no access restriction.

This register can be read and written by privileged and unprivileged access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPMEE
															rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCLR
															rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **IPMEE**: ICACHE and PKA SRAM erase status

This bit is set by hardware when ICACHE and PKA SRAM erase is completed after potential tamper detection (refer to *Section 34: Tamper and backup registers (TAMP)* for more details). This bit is cleared by software by writing 1 to it.

0: ICACHE and PKA SRAM erase on going if not yet cleared by software

1: ICACHE and PKA SRAM erase done

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **MCLR**: Device memories erase status

This bit is set by hardware when SRAM2, ICACHE, and PKA SRAM erase is completed after power-on reset or tamper detection (refer to *Section 34: Tamper and backup registers (TAMP)* for more details). This bit is not cleared by system reset. It is cleared by software by writing 1 to it.

0: Memory erase ongoing if not yet cleared by software

1: Memory erase done

14.3.8 SYSCFG compensation cell control/status register (SYSCFG_CCCSR)

Address offset: 0x01C

Reset value: 0x0000 0002

When the system is secure (TZEN = 1), this register can be protected against nonsecure access by setting the SYSCFGSEC bit in the SYSCFG_SECCFGR register. When the SYSCFGSEC bit is set, only secure access is allowed: nonsecure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN = 0), there is no access restriction.

This register can be read and written by privileged and unprivileged access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDY1	Res.	Res.	Res.	Res.	Res.	Res.	CS1	EN1
							r							rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **RDY1:** VDD I/Os compensation cell ready flag

This bit provides the compensation cell status of the I/Os supplied by V_{DD}.

0: VDD I/Os compensation cell not ready

1: VDD I/Os compensation cell ready

*Note: The HSI16 clock is required for the compensation cell to work properly. The compensation cell ready bit (*RDY1*) is not set if the HSI16 clock is not enabled (HSION).*

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **CS1:** VDD I/Os code selection

This bit selects the code to be applied for the compensation cell of the I/Os supplied by V_{DD}.

0: VDD I/Os code from the cell (available in the SYSCFG_CCVR register)

1: VDD I/Os code from the SYSCFG compensation cell code register (SYSCFG_CCCR)

Bit 0 **EN1:** VDD I/Os compensation cell enable

This bit enables the compensation cell of the I/Os supplied by V_{DD}.

0: VDD I/Os compensation cell disabled

1: VDD I/Os compensation cell enabled

14.3.9 SYSCFG compensation cell value register (SYSCFG_CCVR)

Address offset: 0x020

Reset value: 0x0000 0000

When the system is secure (TZEN = 1), this register can be protected against nonsecure access by setting the SYSCFGSEC bit in the SYSCFG_SECCFGR register. When the SYSCFGSEC bit is set, only secure access is allowed: nonsecure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN = 0), there is no access restriction.

This register can be read and written by privileged and unprivileged access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCV1[3:0]				NCV1[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **PCV1[3:0]:** PMOS compensation value of the I/Os supplied by V_{DD}

This value is provided by the cell and can be used by the CPU to compute an I/Os compensation cell code for PMOS transistors. This code is applied to the I/Os compensation cell when the CS1 bit of the SYSCFG_CCCSR register is cleared.

Bits 3:0 **NCV1[3:0]:** NMOS compensation value of the I/Os supplied by V_{DD}

This value is provided by the cell and can be used by the CPU to compute an I/Os compensation cell code for NMOS transistors. This code is applied to the I/Os compensation cell when the CS1 bit of the SYSCFG_CCCSR is cleared.

14.3.10 SYSCFG compensation cell code register (SYSCFG_CCCR)

Address offset: 0x024

Reset value: 0x0000 0078

When the system is secure (TZEN = 1), this register can be protected against nonsecure access by setting the SYSCFGSEC bit in the SYSCFG_SECCFGR register. When the SYSCFGSEC bit is set, only secure access is allowed: nonsecure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN = 0), there is no access restriction.

This register can be read and written by privileged and unprivileged access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCC1[3:0]				NCC1[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **PCC1[3:0]:** PMOS compensation code of the I/Os supplied by V_{DD}

These bits are written by software to define an I/Os compensation cell code for PMOS transistors. This code is applied to the I/Os compensation cell when the CS1 bit of the SYSCFG_CCCSR is set.

Bits 3:0 **NCC1[3:0]:** NMOS compensation code of the I/Os supplied by V_{DD}

These bits are written by software to define an I/Os compensation cell code for NMOS transistors. This code is applied to the I/Os compensation cell when the CS1 bit of the SYSCFG_CCCSR is set.

14.3.11 SYSCFG flash abort interrupt enable register (SYSCFG_FAIER)

Address offset: 0x110

Reset value: 0x0000 0000

This register can be read and written by privileged access only. An unprivileged read/write access is RAZ/WI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB	Res.	Res.	Res.	WKUP_S	WKUP	RADIO	Res.	SPI3
							rw				rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAI1	I2C3_EV	LPUART1	Res.	USART1	Res.	I2C1_EV	LPDMA_CH3	LPDMA_CH2	LPDMA_CH1	LPDMA_CH0	EXTI5	EXTI4	EXTI3	EXTI2	PVD
rw	rw	rw		rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USB:** enable USB interrupt to abort flash operation.
Access can be secured by GTZC1_TZSC USBSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt.
0: USB interrupt abort flash operation disabled
1: USB interrupt abort flash operation enabled
Note: This bit is reserved on STM32WBA23xx devices.

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **WKUP_S:** enable PWR WKUP secure interrupt to abort flash operation.
Access is always secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt.
0: PWR WKUP secure interrupt abort flash operation disabled
1: PWR WKUP secure interrupt abort flash operation enabled

Bit 19 **WKUP:** enable PWR WKUP interrupt to abort flash operation.
This bit can be access read/write secure and nonsecure.
0: PWR WKUP interrupt abort flash operation disabled
1: PWR WKUP interrupt abort flash operation enabled

Bit 18 **RADIO:** enable RADIO interrupt to abort flash operation.
Access can be secured by GTZC1_TZSC RADIOSEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt.
0: RADIO interrupt abort flash operation disabled
1: RADIO interrupt abort flash operation enabled

Bit 17 Reserved, must be kept at reset value.

Bit 16 **SPI3:** enable SPI3 interrupt to abort flash operation.
Access can be secured by GTZC1_TZSC SPI3SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt.
0: SPI3 interrupt abort flash operation disabled
1: SPI3 interrupt abort flash operation enabled

Bit 15 **SAI1:** enable SAI1 interrupt to abort flash operation.
Access can be secured by GTZC1_TZSC SAI1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt.
0: SAI1 interrupt abort flash operation disabled
1: SAI1 interrupt abort flash operation enabled

Bit 14 **I2C3_EV:** enable I2C3_EV interrupt to abort flash operation.
Access can be secured by GTZC1_TZSC I2C3SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt.
0: I2C3_EV interrupt abort flash operation disabled
1: I2C3_EV interrupt abort flash operation enabled

Bit 13 **LPUART1:** enable LPUART1 interrupt to abort flash operation.
Access can be secured by GTZC1_TZSC LPUART1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt.
0: LPUART1 interrupt abort flash operation disabled
1: LPUART1 interrupt abort flash operation enabled

Bit 12 Reserved, must be kept at reset value.

Bit 11 **USART1:** enable USART1 interrupt to abort flash operation.
Access can be secured by GTZC1_TZSC USART1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt.
0: USART1 interrupt abort flash operation disabled
1: USART1 interrupt abort flash operation enabled

Bit 10 Reserved, must be kept at reset value.

Bit 9 **I2C1_EV:** enable I2C1_EV interrupt to abort flash operation.
Access can be secured by GTZC1_TZSC I2C1SEC. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt.
0: I2C1_EV interrupt abort flash operation disabled
1: I2C1_EV interrupt abort flash operation enabled

Bit 8 **LPDMA_CH3:** enable LPDMA_CH3 interrupt to abort flash operation.
Access can be secured by LPDMA SEC3. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt.
0: LPDMA_CH3 interrupt abort flash operation disabled
1: LPDMA_CH3 interrupt abort flash operation enabled

Bit 7 **LPDMA_CH2:** enable LPDMA_CH2 interrupt to abort flash operation.
Access can be secured by LPDMA SEC2. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt.
0: LPDMA_CH2 interrupt abort flash operation disabled
1: LPDMA_CH2 interrupt abort flash operation enabled

Bit 6 **LPDMA_CH1:** enable LPDMA_CH1 interrupt to abort flash operation.
Access can be secured by LPDMA SEC1. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt.
0: LPDMA_CH1 interrupt abort flash operation disabled
1: LPDMA_CH1 interrupt abort flash operation enabled

Bit 5 **LPDMA_CH0:** enable LPDMA_CH0 interrupt to abort flash operation.
Access can be secured by LPDMA SEC0. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt.
0: LPDMA_CH0 interrupt abort flash operation disabled
1: LPDMA_CH0 interrupt abort flash operation enabled

Bit 4 **EXTI5:** enable EXTI5 interrupt to abort flash operation.
Access can be secured by EXTI SEC5. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt.
0: EXTI5 interrupt abort flash operation disabled
1: EXTI5 interrupt abort flash operation enabled

Bit 3 **EXTI4:** enable EXTI4 interrupt to abort flash operation.
Access can be secured by EXTI SEC4. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt.
0: EXTI4 interrupt abort flash operation disabled
1: EXTI4 interrupt abort flash operation enabled

Bit 2 **EXTI3:** enable EXTI3 interrupt to abort flash operation.

Access can be secured by EXTI SEC3. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt.

0: EXTI3 interrupt abort flash operation disabled

1: EXTI3 interrupt abort flash operation enabled

Bit 1 **EXTI2:** enable EXTI2 interrupt to abort flash operation.

Access can be secured by EXTI SEC2. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt.

0: EXTI2 interrupt abort flash operation disabled

1: EXTI2 interrupt abort flash operation enabled

Bit 0 **PVD:** enable EXTI16 PVD interrupt to abort flash operation.

Access can be secured by EXTI SEC16. When secure, nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt.

0: PVD EXTI16 interrupt abort flash operation disabled

1: PVD EXTI16 interrupt abort flash operation enabled

14.3.12 SYSCFG register map

Table 128. SYSCFG register map and reset values

Offset	Register	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
0x000	SYSCFG_ SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPUSECC	Res.	CLASSBSEC	SYSCFGSEC
	Reset value																													0		0	0
0x004	SYSCFG_CFGR1	RF_EXTPABYP2_INV	RF_EXTPABYP1_INV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PB3_FMP	PA15_FMP	PA7_FMP	PA6_FMP	Res.	Res.	Res.	Res.	Res.	Res.	ANASWVDD	BOOSTEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0											0	0	0	0							0	0								
0x008	SYSCFG_FPUIMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPU_IE[5:0]					
	Reset value																											1	1	1	1	1	1
0x00C	SYSCFG_ CNSLCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKNSMPU	LOCKNSVTOR
	Reset value																															0	0
0x010	SYSCFG_ CSLOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKSAU	LOCKSMPU	LOCKSVTAIRCR
	Reset value																														0	0	0
0x014	SYSCFG_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCL	PVDL	SPL	CLL
	Reset value																													0	0	0	0
0x018	SYSCFG_MESR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPMEE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCLR
	Reset value																0																0
0x01C	SYSCFG_CCCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDY1	Res.	Res.	Res.	Res.	Res.	Res.	CS1	EN1
	Reset value																								0							1	0
0x020	SYSCFG_CCVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCV1[3:0]				NCV1[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0x024	SYSCFG_CCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCC1[3:0]				NCC1[3:0]			
	Reset value																									0	1	1	1	1	0	0	0
0x110	SYSCFG_FAICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB	Res.	Res.	Res.	WKUP_S	WKUP	RADIO	Res.	SPI3	SAI1	I2C3_EV	LPUART1	Res.	USART1	Res.	I2C1_EV	LPDMA_CH3	LPDMA_CH2	LPDMA_CH1	LPDMA_CH0	EXTI5	EXTI4	EXTI3	EXTI2	PVD
	Reset value								0				0	0	0		0	0	0	0		0		0	0	0	0	0	0	0	0	0	0
0x114 to 0x3FC	Reserved																																

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

15 Peripherals interconnect matrix

15.1 Introduction

Several peripherals have direct connections between them, enabling autonomous communication and/or synchronization between them. This saves CPU resources and power consumption. Additionally, these hardware connections remove software latency and result in a more predictable system design.

Depending on the peripherals, these interconnections can operate in Run, Sleep, Stop 0, Stop 1, and Stop 2 modes.

15.2 Connection summary

Table 129. Peripherals interconnect matrix(1) (2)

Source	Destination															
	TIM2	TIM16	TIM17	LPTIM1	LPTIM2	ADC4	LPDMA1	IRTIM	USART1	LPUART1	I2C1	I2C3	SPI3	RTC	TAMP	AES
TIM2	-	-	-	-	-	*2*	*8*	-	-	-	-	-	-	-	-	-
TIM16	*1*	-	-	-	-	-	-	*12*	-	-	-	-	-	-	-	-
TIM17	*1*	-	-	-	-	-	-	*12*	-	-	-	-	-	-	-	-
LPTIM1	-	-	-	-	-	*2*	*8*	-	*10*	*10*	*10*	*10*	*10*	-	-	-
LPTIM2	-	-	-	-	-	-	*8*	-	*10*	-	*10*	-	-	-	-	-
ADC4	*3*	-	-	-	-	-	*8*	-	-	-	-	-	-	-	*9*	-
Temperature sensor	-	-	-	-	-	*6*	-	-	-	-	-	-	-	-	-	-
V_{CORE}	-	-	-	-	-	*6*	-	-	-	-	-	-	-	-	-	-
VREFINT	-	-	-	-	-	*6*	-	-	-	-	-	-	-	-	-	-
HSE32	-	*4*	*4*	-	-	-	-	-	-	-	-	-	-	-	-	-
HSI16	*4*	*4*	*4*	-	*4*	-	-	-	-	-	-	-	-	-	-	-
LSE	*4*	*4*	*4*	*4*	-	-	-	-	-	-	-	-	-	-	*9*	-
LSI	-	*4*	*4*	*4*	-	-	-	-	-	-	-	-	-	-	-	-
MCO	-	*4*	*4*	-	-	-	-	-	-	-	-	-	-	-	-	-
GPIO EXTI	-	-	-	*5*	*5*	*2*	*8*	-	*10*	*10*	*10*	*10*	*10*	-	-	-
RTC	-	-	-	*5*	*5*	-	*8*	-	*10*	*10*	*10*	*10*	*10*	-	*9*	-
TAMP	-	-	-	-	*5*	-	*8*	-	-	-	-	-	-	*11*	-	-
LPDMA1	-	-	-	-	*5*	-	*8*	-	*10*	*10*	*10*	*10*	*10*	-	-	-
SYST ERR	-	*7*	*7*	-	-	-	-	-	-	-	-	-	-	-	-	-
AES	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*9*	-

Table 129. Peripherals interconnect matrix(1) (2) (continued)

Source	Destination															
	TIM2	TIM16	TIM17	LPTIM1	LPTIM2	ADC4	LPDMA1	IRTIM	USART1	LPUART1	I2C1	I2C3	SPI3	RTC	TAMP	AES
PKA	-	-	-	-	-	-	-	-	-	-	-	-	-	-	9	-
TRNG	-	-	-	-	-	-	-	-	-	-	-	-	-	-	9	-
IWDG	-	-	-	-	-	-	-	-	-	-	-	-	-	-	9	-
DEBUG	-	-	-	-	-	-	-	-	-	-	-	-	-	-	9	-

1. The numbers in this table are links to the corresponding subsections of *Section 15.3*.
2. The "-" symbol in grayed cells means no interconnect.

15.3 Interconnection details

15.3.1 Master to slave interconnection for timers

From timer (TIM2/TIM16/TIM17) to timer (TIM2).

Purpose

Some timers are linked together internally for timer synchronization or chaining.

When one timer is configured in master mode, it can reset, start, stop, or clock the counter of another timer configured in slave mode.

The synchronization modes are detailed in:

- *Section 28.4.23: Timer synchronization* and *Section 28.4.22: Timers and external trigger synchronization* for general-purpose timers (TIM2)

Triggering signals

The output from the master timer is on signal tim_trgo for TIM2, and tim_oc for TIM16 and TIM17, following a configurable timer event. The input to the slave timer is on signals tim_itr.

The possible master/slave connections are given in:

- *Table 255: TIMx internal trigger connection* for TIM2

Active power modes

Run and Sleep.

15.3.2 Triggers to ADC4

From EXTI and timers (TIM2) and (LPTIM1) to ADC4.

Purpose

The timers TIM2 can be used to generate the ADC4 trigger event through the timer outputs tim_oc or tim_trgo. The low-power timer LPTIM1 can be used to generate the ADC4 trigger

event through output lptim1_ch1. In addition, GPIO pin 15 via EXTI channel can be used to generate an ADC4 trigger event.

Triggering signals

The input trigger signal list and the description of the interconnection between ADC4 and timers and EXTI is given in:

- *Table 167: ADC interconnection*
- *Section 22.4.16: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN)*.
- *Section 22.4.21: Example timing diagrams (single/continuous modes hardware/software triggers)*

Active power modes

Run, Sleep, and for LPTIM also in Stop 0, 1, and 2 modes, and EXTI in all Stop modes.

15.3.3 ADC4 analog watchdog as trigger to timer

From ADC4 to TIM2.

Purpose

The internal analog watchdog output signals from ADC4 are connected to timers. ADC4 can provide trigger events through watchdog signals to the timer (TIM2) to reset, start, stop, or enable the counting.

Setting descriptions of the ADC analog watchdog and timer trigger are provided in:

- *Section 28.4.2: TIM2 pins and internal signals* for TIM2
- *Table 256: Interconnect to the tim_etr input multiplexer* for TIM2

Triggering signals

The output from ADC4 is on signals adc_awd (three watchdogs on ADC4) and the input to timer on signal tim_etr.

Active power modes

Run, Sleep, and an ADC4 conversion in autonomous mode in Stop 0, 1, and 2 modes can generate a wake-up interrupt and desired trigger action to timers.

15.3.4 Clock source to timer

From HSE32, HSI16, LSE, LSI, and MCO to timer (TIM2/TIM16/TIM17) and low-power timer (LPTIM1/LPTIM2).

Purpose

A timer input or clock can receive different clock sources and can be used, for example, to calibrate internal oscillators and a reference clock.

External clocks (HSE32, LSE), internal clocks (HSI16, LSI), and microcontroller output clock (MCO) can be used as input to timer.

- HSI16 and LSE are assigned to timer TIM2 as external input signals (tim_etr4/tim_etr11). HSI/LSE can be selected as a counter clock provided by

an external clock source in mode2: external trigger input. Inputs assignment and clock selection description are detailed in *Section 28.4.5: Clock selection*.

- HSE32, HSI16, LSE, and LSI are assigned to general-purpose timers TIM16 and TIM17 as an input multiplexer input signal (tim_ti1_in3/tim_ti1_in5/tim_ti1_in6/tim_ti1_in9). HSI16/LSE/LSI can be selected as a counter clock provided by an external clock source in mode1: input multiplexer input. Inputs assignment and clock selection description are detailed in *Section 28.4.5: Clock selection*.
- MCO is connected as external input to general-purpose timers TIM16 and TIM17, making possible the calibration of the HSI16 system clock with LSE or LSI with HSE system clock. This feature is detailed in *Section 28.4.5: Clock selection*.
- LSE and LSI can be selected as input capture 2 (lptim_ic2_mux1/lptim_ic2_mux2) to LPTIM1. This feature is detailed in *Section 28.4.7: Input capture mode*.
- HSI16/256 can be selected as input capture 2 (lptim_ic2_mux1) to LPTIM2. This feature is detailed in *Section 28.4.7: Input capture mode*.

Triggering signals

The input to the timer is on signals tim_etr or tim_ti1_in, and for low-power timer on signals lptim_ic2_mux.

The possible connections are given in:

- *Table 256: Interconnect to the tim_etr input multiplexer* for TIM2
- *Table 251: Interconnect to the tim_ti1 input multiplexer* for TIM16 and TIM17
- *Table 287: LPTIM1/2 input 2 connections* for LPTIM1/LPTIM2

Active power modes

Run and Sleep.

15.3.5 Triggers to low-power timer

From RTC wake-up, RTC alarm, TAMP, LPDMA1, and LPTIM_ETR to low-power timer (LPTIM1/LPTIM2).

Purpose

Low-power timer counters may be started after the detection of an active edge on a trigger input (lptim_ext_trig0 to 5). This feature is detailed in *Section 30.4.7: Trigger multiplexer*.

Triggering signals

The input to low-power timer on signals lptim_ext_trig.

The possible connections are given in *Table 285: LPTIM1/2 external trigger connections*.

Active power modes

Run, Sleep, and Stop 0, and for RTC and TAMP also in Stop 1 and Stop 2 modes.

15.3.6 Internal analog signals to analog peripheral

From internal analog source to analog peripheral (ADC4).

Purpose

The internal reference voltage (V_{REFINT}), the internal temperature sensor (V_{SENSE}), and digital core voltage (V_{CORE}) monitoring signals are connected to analog peripheral (ADC4), as described in:

- *Section 22.4.9: Channel selection (CHSEL, SCANDIR, CHSELRMOD)*
- *Section 22.4.27: Temperature sensor and internal reference voltage*

Input signals

The input to analog peripheral on signals V_{IN}.

The possible connections are given in *Table 167: ADC interconnection*.

Active power modes

Run, Sleep, Stop 0, and Stop 1.

15.3.7 System errors as break signals to timers

From system errors to timers (TIM16/TIM17).

Purpose

HSE32 clock security, CPU lockup, SRAM2 parity error, FLASH ECC double error detection, and PVD can generate system errors in the form of timer break toward timers (TIM16/TIM17).

The purpose of the break function is to protect power switches driven by PWM signals generated by the timers. This feature is detailed in:

- *Section 29.4.13: Using the break function* for TIM16 and TIM17

Break signals

The input to the timer on signals tim_sys_brk.

The possible connections are given in:

- *Table 271: Timer break interconnect* for TIM16 and TIM17

Active power modes

Run and Sleep.

15.3.8 Triggers to LPDMA1

From GPIO pin EXTI, RTC, TAMP, timers (TIM2), low-power timer (LPTIM1/LPTIM2), LPDMA1, ADC4 to LPDMA1.

Purpose

An LPDMA trigger can be assigned to an LPDMA channel x. A programmed LPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event. The trigger mode can also be programmed to condition the linked-list item transfer.

More details are given in:

- *Section 16.3.5: LPDMA triggers*
- *Section 16.4.12: LPDMA triggered transfer*

Triggering signals

Trigger mapping is specified in *Table 16.3.5: LPDMA triggers*.

Active power modes

Run and Sleep, and for peripherals supporting autonomous mode also in Stop 0 and Stop 1.

15.3.9 Internal tamper sources

From LSE clock security, RTC, Debug, ADC4, AES, PKA, TRNG, and IWDG to TAMP

Purpose

To detect any abnormal activity or tentative to corrupt the device, tampers are introduced and alert the system to such undesired events. Different actions can be taken in consequence. More details are given in *Section 34: Tamper and backup registers (TAMP)*.

Resources

Table 314: TAMP interconnection presents a list of tamper sources.

Active power modes

These interconnections are active in all power modes if the tamper source is active.

15.3.10 Triggers to communication peripherals

From LP timers (LPTIM1/LPTIM2), LPDMA1 transfer complete, EXTI GPIOs, RTC alarm and RTC wake-up to I2C1, I2C3, USART1, LPUART1, and SPI3.

Purpose

LP timer (LPTIM1) output channels (lptim1_ch1 and lptim2_ch1), EXTI GPIOs, RTC alarm and RTC wake-up, can be used as trigger to start a communication on the selected I2C, USART, LPUART, and SPI peripheral.

An LPDMA1 transfer complete can trigger both the LPDMA1 regular or linked-list new transfers and communication on the selected communication peripheral.

These features are detailed in:

- *Section 35.4.16: Autonomous mode* I2C
- *Section 36.5.22: USART autonomous mode* USART
- *Section 37.4.15: LPUART autonomous mode* LPUART
- *Section 38.4.15: Autonomous mode* SPI

Triggering signals

The outputs from triggers are directly connected to peripheral trigger inputs.

The selection of input triggers is detailed in:

- *Table 326: I2C1 interconnection* and *Table 327: I2C3 interconnection*
- *Table 346: USART interconnection (USART1)*
- *Table 358: LPUART interconnections (LPUART1)*
- *Table 368: SPI interconnection (SPI3)*

Active power mode

These interconnections remain active in Run, Sleep, and Stop modes if both the source and communication line are autonomous in this mode. Refer to:

- *Section 36.6: USART in low-power modes*
- *Section 35.5: I2C in low-power modes*
- *Section 38.6: SPI in low-power modes*

15.3.11 Output from tamper

From TAMP to RTC.

Purpose

The RTC can timestamp a tamper event to retrieve history in time of such detection. The RTC can also control RTC_OUT and send tamp status tamp_evt outside the MCU. More details are given in *Section 33.3.3: GPIOs controlled by the RTC and TAMP*.

Active power mode

This interconnection remains active in all power modes.

15.3.12 Timers generating IRTIM signal

From timer (TIM16/TIM17) to IRTIM.

Purpose

Timers (TIM16/TIM17) output channels timx_oc1 are used to generate the waveform of infrared signal output IRTIM. The functionality is detailed in *Section 31: Infrared interface (IRTIM)*.

Active power mode

Run and Sleep.

16 Low-power direct memory access controller (LPDMA)

16.1 LPDMA introduction

The low-power direct memory access (LPDMA) controller is a bus master and system peripheral.

The LPDMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories via linked-lists, upon the control of an off-loaded CPU.

16.2 LPDMA main features

- Single bidirectional AHB master
- Memory-mapped data transfers from a source to a destination:
  - Peripheral-to-memory
  - Memory-to-peripheral
  - Memory-to-memory
  - Peripheral-to-peripheral
- Autonomous data transfers during Stop mode (see *Section 16.4.17*)
- Transfers arbitration based on a 4-grade programmed priority at channel level:
  - One high-priority traffic class, for time-sensitive channels (queue 3)
  - Three low-priority traffic classes, with a weighted round-robin allocation for non time-sensitive channels (queues 0, 1, 2)
- Per channel event generation, on any of the following events: transfer complete, half-transfer complete, data transfer error, user setting error, link transfer error, completed suspension and trigger overrun
- Per channel interrupt generation, with separately programmed interrupt enable per event
- 8 concurrent LPDMA channels:
  - Intra-channel LPDMA transfers chaining via programmable linked-list into memory, supporting two execution modes: run-to-completion and link step mode
  - Intra-channel and inter-channel LPDMA transfers chaining via programmable LPDMA input triggers connection to LPDMA task completion events
- Per linked-list item within a channel:
  - Separately programmed source and destination transfers
  - Programmable data handling between source and destination: byte-based padding or truncation, sign extension and left/right realignment
  - Programmable number of data bytes to be transferred from the source, defining the block level
  - Linear source and destination addressing: either fixed or contiguously incremented addressing, programmed at a block level, between successive single transfers
  - Programmable LPDMA request and trigger selection
  - Programmable LPDMA half-transfer and transfer-complete events generation

  - Pointer to the next linked-list item and its data structure in memory, with automatic update of the LPDMA linked-list control registers
- Debug:
  - Channel suspend and resume support
  - Channel status reporting and event flags
- TrustZone support
  - Support for secure and nonsecure LPDMA transfers, independently at a first channel level, and independently at a source/destination and link sub-levels
  - Secure and nonsecure interrupts reporting, resulting from any of the respectively secure and nonsecure channels
  - TrustZone-aware AHB slave port, protecting any LPDMA secure resource (register, register field) from a nonsecure access
- Privileged/unprivileged support:
  - Support for privileged and unprivileged LPDMA transfers, independently at a channel level
  - Privileged-aware AHB slave port

16.3 LPDMA implementation

16.3.1 LPDMA channels

A given LPDMA channel x is implemented with the hardware parameters as per the table below.

Table 130.LPDMA1 channels implementation

Channel x	Hardware parameters		Features
	dma_fifo_ size[x]	dma_ addressing[x]	
x = 0 to 7	0	0	Channel x (x = 0 to 7) is implemented with: – no FIFO. Only a single source transfer cell is internally registered. – fixed/contiguously incremented addressing

16.3.2 LPDMA autonomous mode in low-power modes

The LPDMA autonomous mode and wake-up features are implemented in the device low-power modes as per the table below.

Table 131. LPDMA1 autonomous mode and wake-up in low-power modes

Feature	Low-power modes
Autonomous mode and wake-up	LPDMA1 in Sleep, Stop 0, and Stop 1 modes(1)

1. LPDMA is powered down in Stop 2 and Stop 3 modes.

16.3.3 LPDMA requests

A LPDMA request from a peripheral can be assigned to a LPDMA channel x, via the REQSEL[5:0] field in the *LPDMA channel x transfer register 2 (LPDMA_CxTR2)*, provided that SWREQ = 0.

The LPDMA requests mapping is specified in the table below.

Table 132. Programmed LPDMA1 request

LPDMA_CxTR2.REQSEL[5:0]	Selected LPDMA request
0	adc4_dma
1	Reserved
2	Reserved
3	spi3_rx_dma
4	spi3_tx_dma
5	i2c1_rx_dma
6	i2c1_tx_dma
7	i2c1_evc_dma
8	i2c3_rx_dma
9	i2c3_tx_dma
10	i2c3_evc_dma
11	usart1_rx_dma
12	usart1_tx_dma
13	Reserved
14	Reserved
15	lpuart1_rx_dma
16	lpuart1_tx_dma
17	sai_a_dma
18	sai_b_dma
19	Reserved
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	tim2_cc1_dma
27	tim2_cc2_dma
28	tim2_cc3_dma
29	tim2_cc4_dma

Table 132. Programmed LPDMA1 request (continued)

LPDMA_CxTR2.REQSEL[5:0]	Selected LPDMA request
30	tim2_upd_dma
31	tim16_cc1_dma
32	tim16_upd_dma
33	tim17_cc1_dma
34	tim17_upd_dma
35	aes_in_dma
36	aes_out_dma
37	hash_in_dma
38	lptim1_ic1_dma
39	lptim1_ic2_dma
40	lptim1_ue_dma
41	lptim2_ic1_dma
42	lptim2_ic2_dma
43	lptim2_ue_dma
44	xspi1_dma(1)

1. Only available on STM32WBA25xx devices.

16.3.4 LPDMA block requests

Some LPDMA requests must be programmed as a block request, and not as a single request. Then the BREQ bit in *LPDMA channel x transfer register 2 (LPDMA_CxTR2)* must be set for a correct LPDMA execution of the requested peripheral transfer at the hardware level.

The LPDMA block requests are listed in the table below.

Table 133. Programmed LPDMA1 request as a block request

LPDMA block requests
lptim1_ue_dma
lptim2_ue_dma

16.3.5 LPDMA triggers

A LPDMA trigger can be assigned to a LPDMA channel x, via the TRIGSEL[4:0] field in the *LPDMA channel x transfer register 2 (LPDMA_CxTR2)*, provided that TRIGPOL[1:0] defines a rising or a falling edge of the selected trigger (TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

The LPDMA triggers mapping is specified in the table below.

Table 134.Programmed LPDMA1 trigger

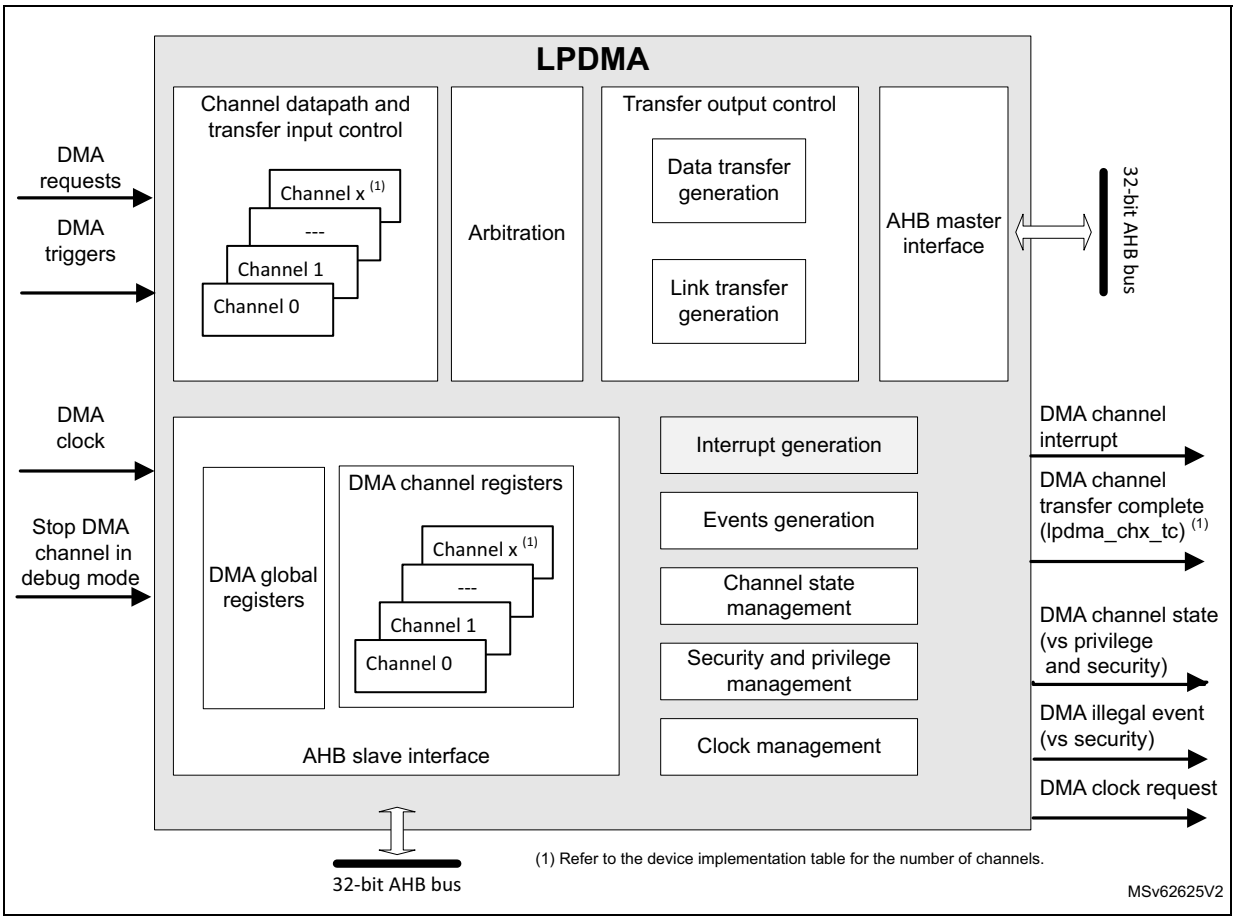
LPDMA_CxTR2.TRIGSEL[4:0]	Selected LPDMA trigger
0	exti0
1	exti1
2	exti2
3	exti3
4	exti4
5	exti5
6	exti6
7	exti7
8	tamp_trg1
9	tamp_trg2
10	tamp_trg3
11	lptim1_ch1
12	lptim1_ch2
13	lptim2_ch1
14	lptim2_ch2
15	rtc_alra_trg
16	rtc_alrb_trg
17	rtc_wut_trg
18	lpdma1_ch0_tc
19	lpdma1_ch1_tc
20	lpdma1_ch2_tc
21	lpdma1_ch3_tc
22	lpdma1_ch4_tc
23	lpdma1_ch5_tc
24	lpdma1_ch6_tc
25	lpdma1_ch7_tc
26	tim2_trgo
27	adc4_awd1

16.4 LPDMA functional description

16.4.1 LPDMA block diagram

The LPDMA block diagram is illustrated in the figure below.

Figure 48. LPDMA block diagram



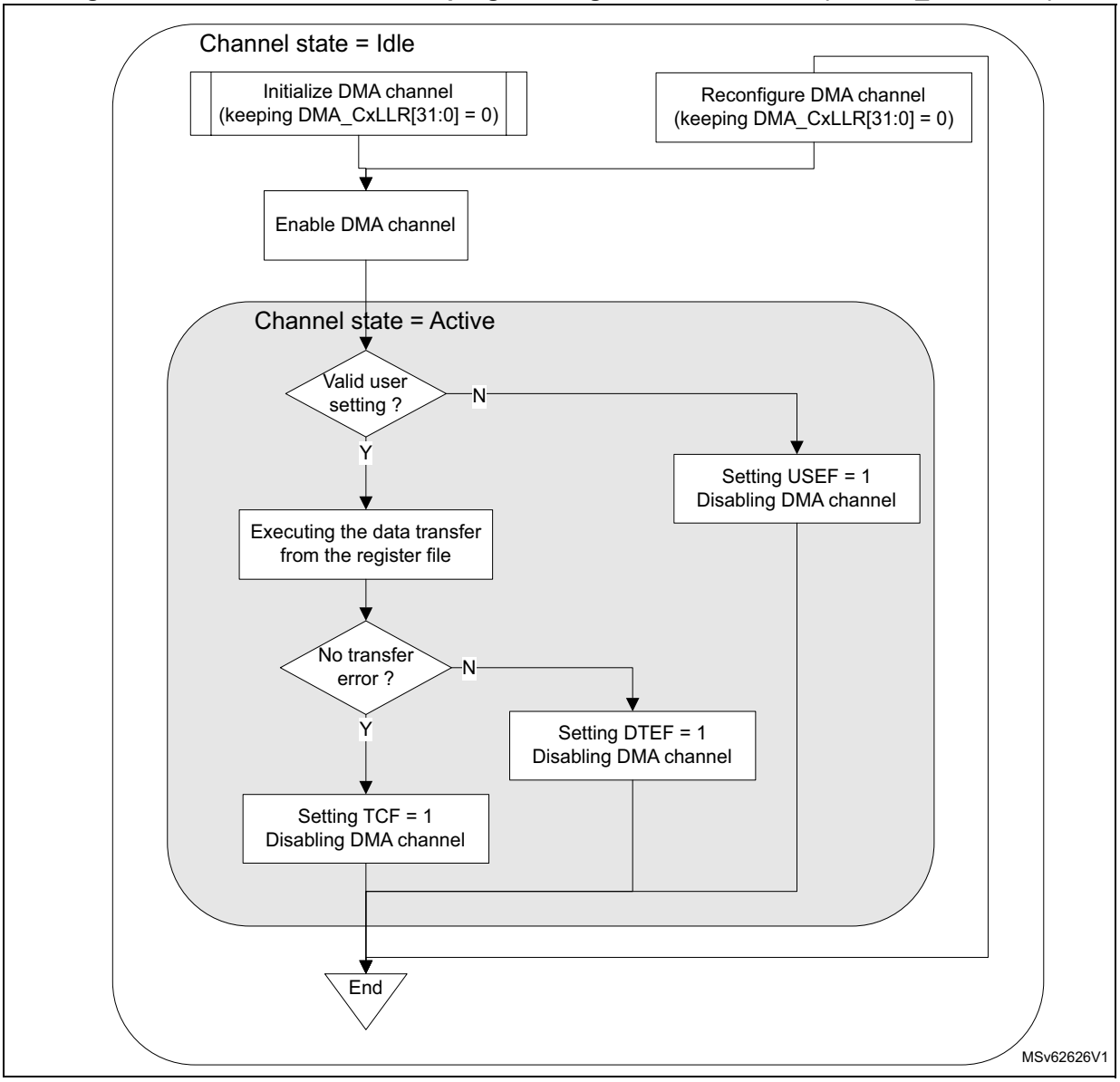
16.4.2 LPDMA channel state and direct programming without any linked-list

After a LPDMA reset, a LPDMA channel x is in idle state. When the software writes 1 into the in LPDMA_CxCR.EN enable control bit, the channel takes into account the value of the different channel configuration registers (LPDMA_CxXXX), switches to the active/non-idle state and starts to execute the corresponding requested data transfers.

After enabling/starting a LPDMA channel transfer by writing 1 into the LPDMA_CxCR.EN bit, a LPDMA channel interrupt on a complete transfer notifies the software that the LPDMA channel is back in idle state (EN is then de-asserted by hardware) and that the channel is ready to be reconfigured then enabled again.

The figure below illustrates this LPDMA direct programming without any linked-list (LPDMA_CxLLR = 0).

Figure 49. LPDMA channel direct programming without linked-list (LPDMA_CxLLR = 0)



16.4.3 LPDMA channel suspend and resume

The software can suspend on its own a channel still active, with the following sequence:

1. The software writes 1 into the LPDMA_CxCR.SUSP bit.
2. The software polls the suspended flag LPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to LPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of

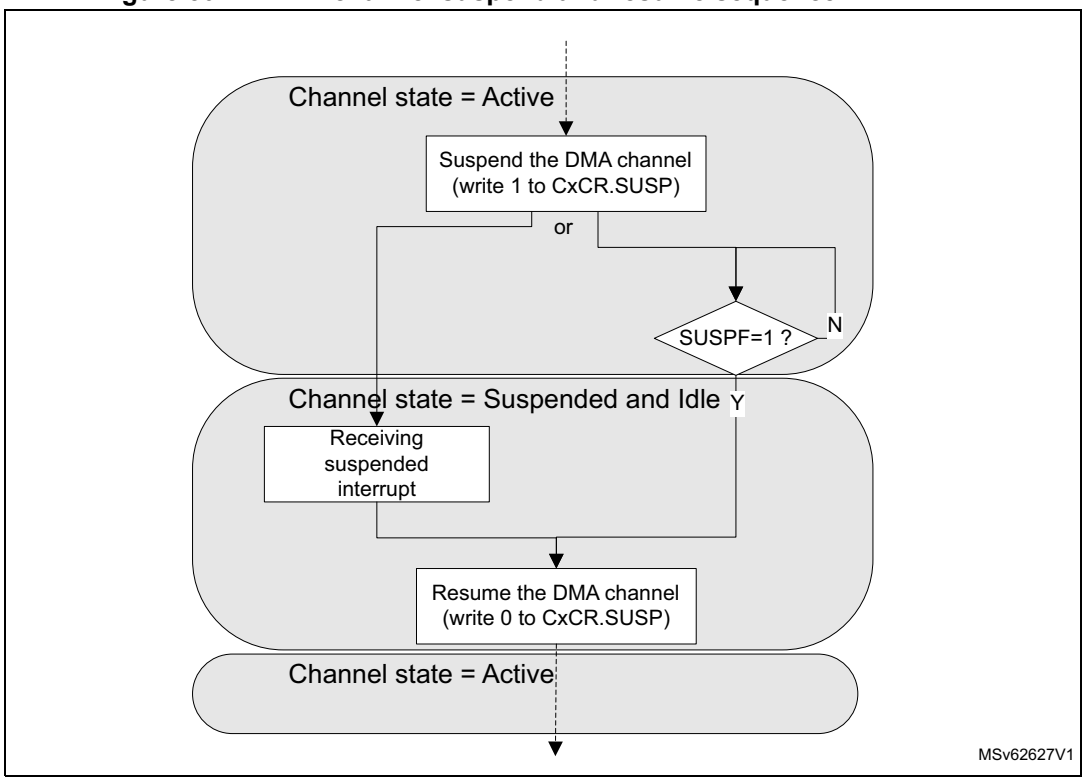
any ongoing LPDMA transfer over its master port. Then the software can observe, in a steady state, any read register or register field that is hardware modifiable.

Note that an ongoing LPDMA transfer can be a data transfer (a read-followed-by-write single transfer) or a link transfer for the internal update of the linked-list register file from the next linked-list item.

3. The software safely resumes the suspended channel by writing 0 to LPDMA_CxCR.SUSP.

The suspend and resume sequence is illustrated in the figure below.

Figure 50. LPDMA channel suspend and resume sequence



Note: A suspend and resume sequence does not impact the LPDMA_CxCR.EN bit. Suspending a channel (transfer) does not suspend a started trigger detection.

16.4.4 LPDMA channel abort and restart

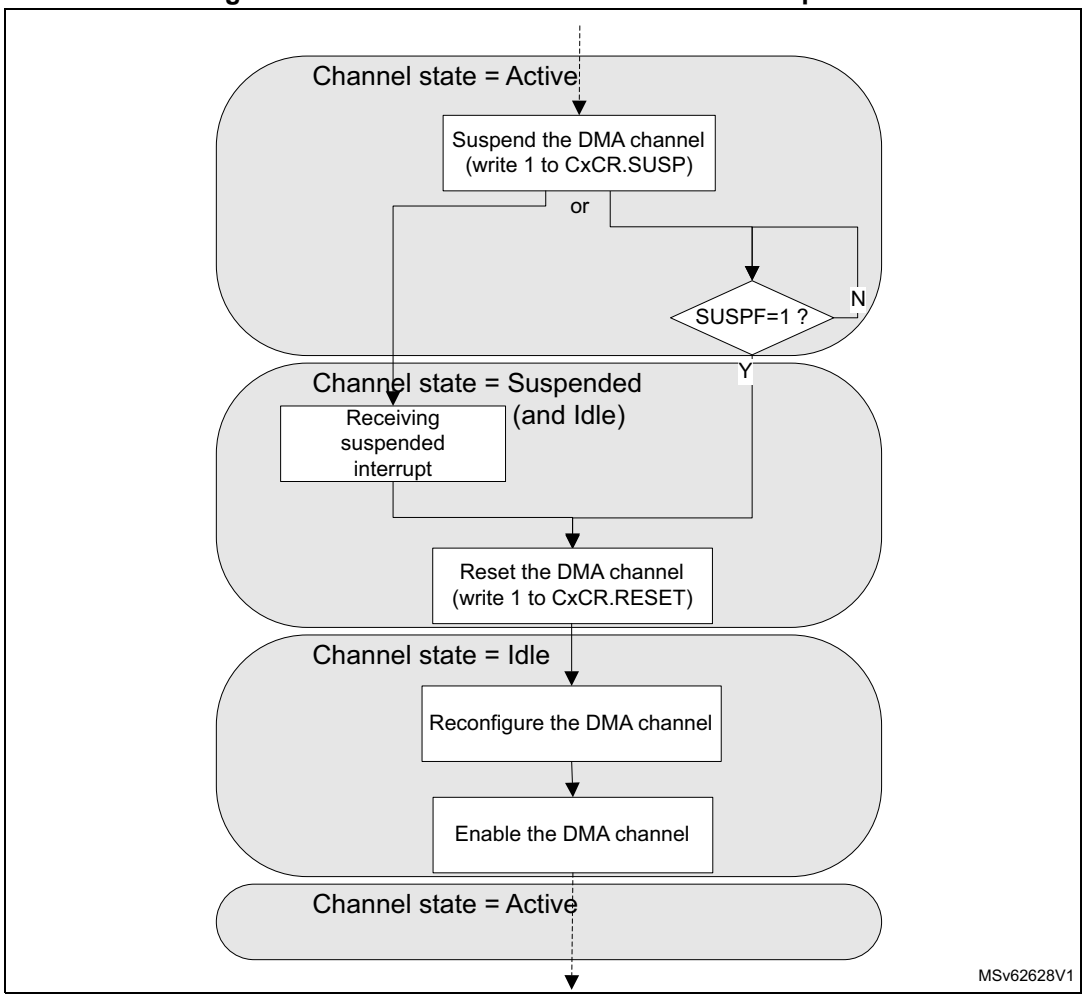
Alternatively, like for aborting a continuous LPDMA transfer with a circular buffering or a double buffering, the software can abort, on its own, a still active channel with the following sequence:

1. The software writes 1 into the LPDMA_CxCR.SUSP bit.
2. The software polls suspended flag LPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to LPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing LPDMA transfer over its master port.
3. The software resets the channel by writing 1 to LPDMA_CxCR.RESET. This causes the reset of the channel internal state, the reset of the LPDMA_CxCR.EN bit, and the reset of the LPDMA_CxCR.SUSP bit.

4. The software safely reconfigures the channel. The software must reprogram the hardware-modified LPDMA_CxBR1, LPDMA_CxSAR, and LPDMA_CxDAR registers.
5. In order to restart the aborted then reprogrammed channel, the software enables it again by writing 1 to the LPDMA_CxCR.EN bit.

The abort and restart sequence is illustrated in the figure below.

Figure 51. LPDMA channel abort and restart sequence



16.4.5 LPDMA linked-list data structure

Alternatively to the direct programming mode, a channel can be programmed by a list of transfers, known as a list of linked-list items (LLI). Each LLI is defined by its data structure.

The base address in memory of the data structure of a next LLI_{n+1} of a channel x is the sum of the following:

- the link base address of the channel x (in LPDMA_CxLBAR)
- the link address offset (LA[15:2] field in LPDMA_CxLLR)

The data structure for each LLI may be specific.

A linked-list data structure is addressed following the value of the UT1, UT2, UB1, USA, UDA and ULL bits of the LPDMA_CxLLR register.

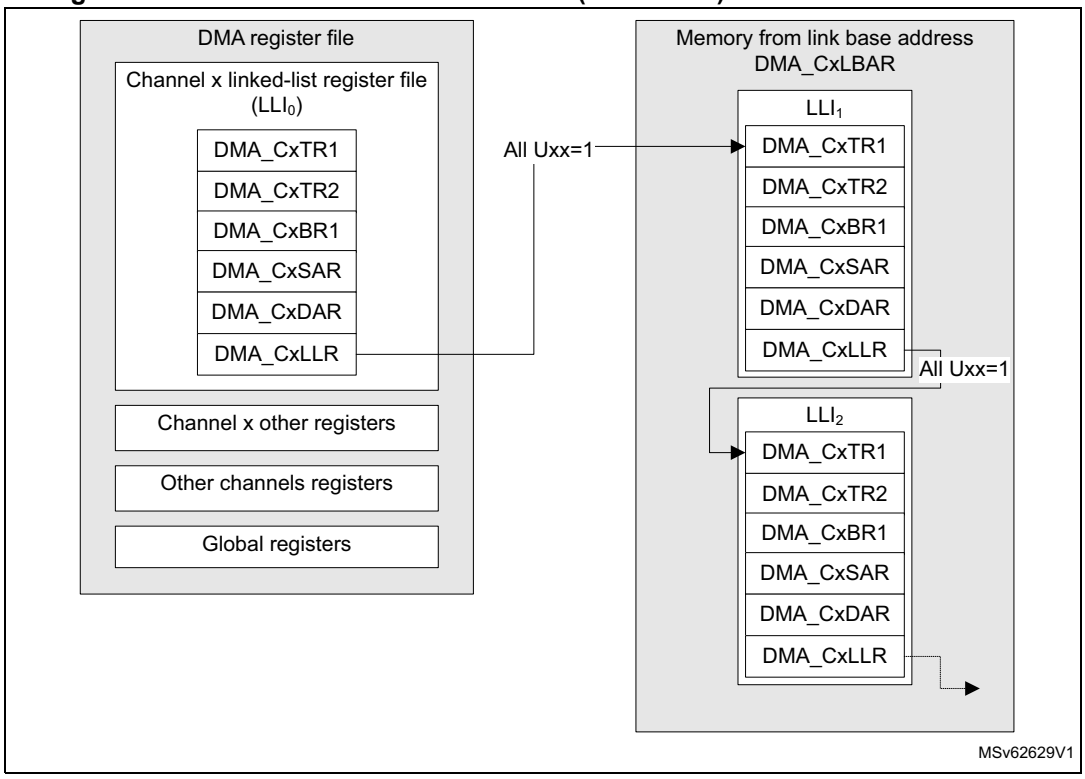
In linked-list mode, each LPDMA linked-list register (LPDMA_CxTR1, LPDMA_CxTR2, LPDMA_CxBR1, LPDMA_CxSAR, LPDMA_CxDAR or LPDMA_CxLLR) is conditionally and automatically updated from the next linked-list data structure in the memory, following the current value of the LPDMA_CxLLR register that was conditionally updated from the linked-list data structure of the previous LLI.

Caution: The user must program the pointer to the next linked-list data structure (LPDMA_CxLLR[15:0]) not to exceed the 64-Kbyte addressable space defined by the link base address register (LPDMA_CxLBAR). The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Static linked-list data structure

For example, when the update bits (UT1, UT2, UB1, USA, UDA and ULL) of the LPDMA_CxLLR register are all asserted, the linked-list data structure in memory is maximal with six contiguous 32-bit locations, including LPDMA_CxTR1, LPDMA_CxTR2, LPDMA_CxBR1, LPDMA_CxSAR, LPDMA_CxDAR and LPDMA_CxLLR (see the figure below) and including the first linked-list register file (LLI_0) and the next LLIs (LLI_1, LLI_2,...) in the memory.

Figure 52. Static linked-list data structure (all Uxx = 1) of channel x



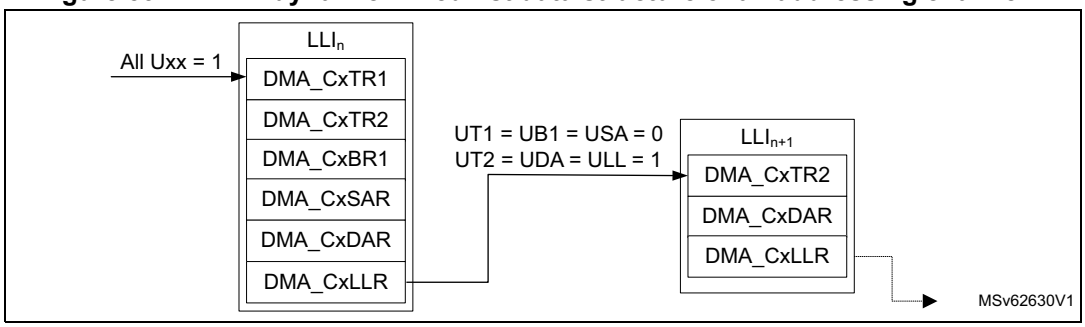
Dynamic linked-list data structure

Alternatively, the memory organization for the full list of LLIs can be compacted with specific data structure for each LLI.

If UT1 = 0 and UT2 = 1, the link address offset of the register LPDMA_CxLLR is pointing to the updated value of the LPDMA_CxTR2 instead of the LPDMA_CxTR1 which is not to be modified.

Example: if UT1 = UB1 = USA = 0, and if UT2 = UDA = ULL = 1, the next LLI does not contain an updated value for LPDMA_CxTR1, nor LPDMA_CxBR1, nor LPDMA_CxSAR. The next LLI contains an updated value for LPDMA_CxTR2, LPDMA_CxDAR, and LPDMA_CxLLR, as shown in the figure below.

Figure 53. LPDMA dynamic linked-list data structure of an addressing channel x



16.4.6 Linked-list item transfer execution

A LLI_n transfer is the sequence of:

1. a data transfer: LPDMA executes the data transfer as described by the LPDMA internal register file (this data transfer can be void/null for LLI_0)
2. a conditional link transfer: LPDMA automatically and conditionally updates its internal register file by the data structure of the next LLI_{n+1}, as defined by the LPDMA_CxLLR value of the LLI_n.

Note: The initial data transfer as defined by the internal register file (LLI_0) can be null (LPDMA_CxBR1.BNDT[15:0] = 0) provided that the conditional update bit UB1 in LPDMA_CxLLR is set (meaning there is a non-null data transfer described by the next LLI_1 in the memory to be executed).

Depending on the intended LPDMA usage, a LPDMA channel x can be executed as described by the full linked-list (run-to-completion mode, LPDMA_CxCR.LSM = 0) or a LPDMA channel x can be programmed for a single execution of a LLI (link step mode, LPDMA_CxCR.LSM = 1), as described in the next paragraphs.

16.4.7 LPDMA channel state and linked-list programming in run-to-completion mode

When LPDMA_CxCR.LSM = 0, a LPDMA channel x is initially programmed, started by writing 1 to LPDMA_CxCR.EN, and after (possibly) completed at channel level. The channel transfer is:

- configured with at least the following:
  - the first LLI_0, internal linked-list register file: LPDMA_CxTR1, LPDMA_CxTR2, LPDMA_CxBR1, LPDMA_CxSAR, LPDMA_CxDAR and LPDMA_CxLLR
  - the last LLI_N, described by the linked-list data structure in memory, as defined by the LPDMA_CxLLR reflecting the before last LLI_{N-1}
- completed when LPDMA_CxLLR[31:0] = 0 and LPDMA_CxBR1.BNDT[15:0] = 0, at the end of the last LLI_{N-1} transfer

LPDMA_CxLLR[31:0] = 0 is the condition of a linked-list based channel completion and means the following:

- The 16 low significant bits LPDMA_CxLLR.LA[15:0] of the next link address are null.
- All the update bits Uxx of LPDMA_CxLLR are null (UT1, UT2, UB1, USA, UDA, ULL).

The channel may never be completed when LPDMA_CxLLR.LSM = 0:

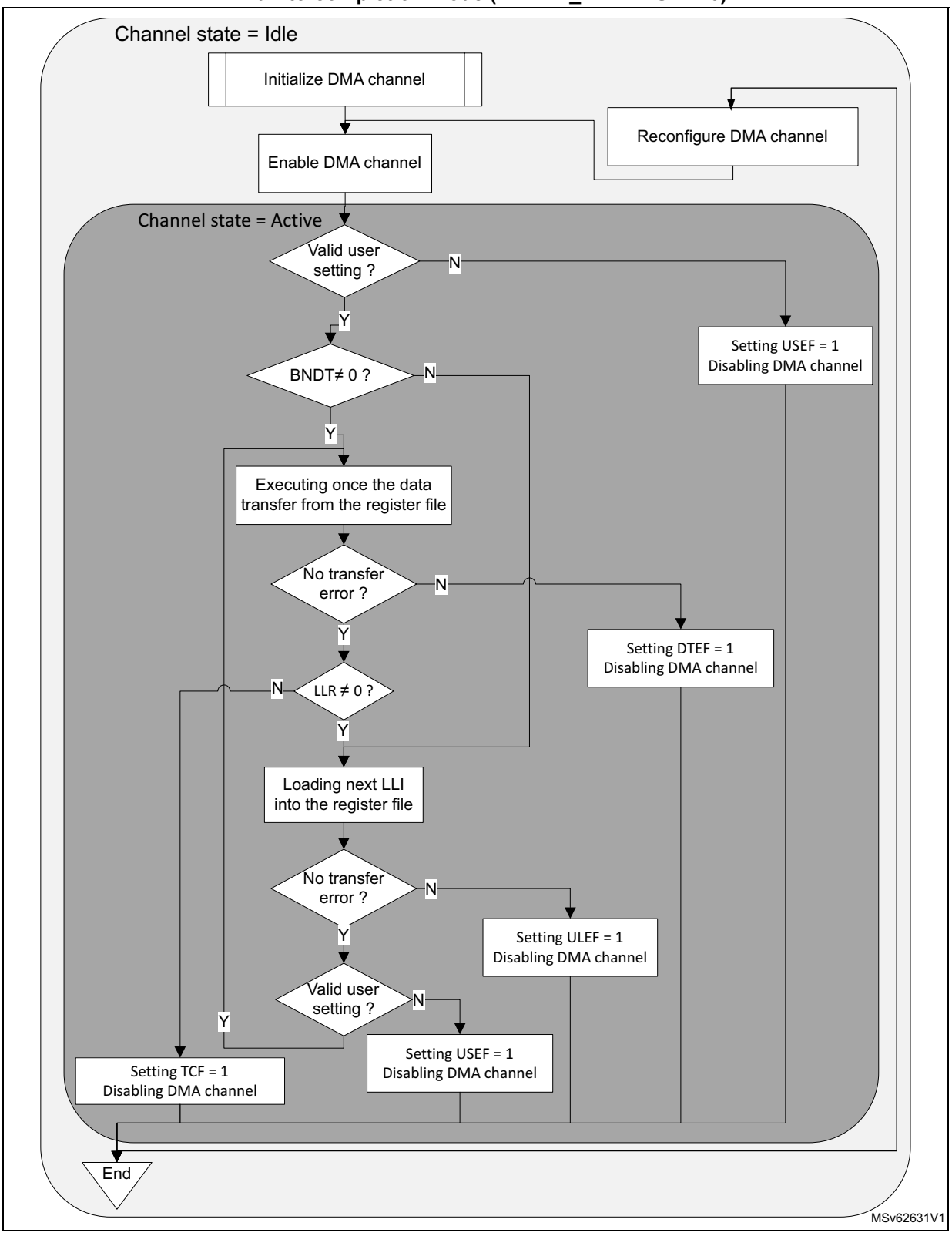
- If the last LLI_N is recursive, pointing to itself as a next LLI:
  - either LPDMA_CxLLR.ULL = 1 and LPDMA_CxLLR.LA[15:2] is updated with the same value
  - or LPDMA_CxLLR.ULL = 0
- If LLI_N is pointing to a previous LLI.

In the typical run-to-completion mode, the allocation of a LPDMA channel, including its fine programming, is done once during the LPDMA initialization. In order to have a reserved data communication link and LPDMA service during run-time, for continuously repeated transfers (from/to a peripheral respectively to/from memory or for memory-to-memory transfers). This reserved data communication link can consist of a channel, or the channel can be shared and a repeated transfer consists of a sequence of LLIs.

Figure 54 depicts the LPDMA channel execution and its registers programming in run-to-completion mode.

Note: Figure 54 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in LPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at channel completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion, the hardware always set TCEF = 1 and disables the channel.

Figure 54. LPDMA channel execution and linked-list programming in run-to-completion mode (LPDMA_CxCR.LSM = 0)



Run-time inserting a LLI_n via an auxiliary channel, in run-to-completion mode

The start of the link transfer of the LLI_{n-1} (start of the LLI_n loading) can be conditioned by the occurrence of a trigger, when programming the following fields of the LPDMA_CxTR2 in the data structure of the LLI_{n-1}:

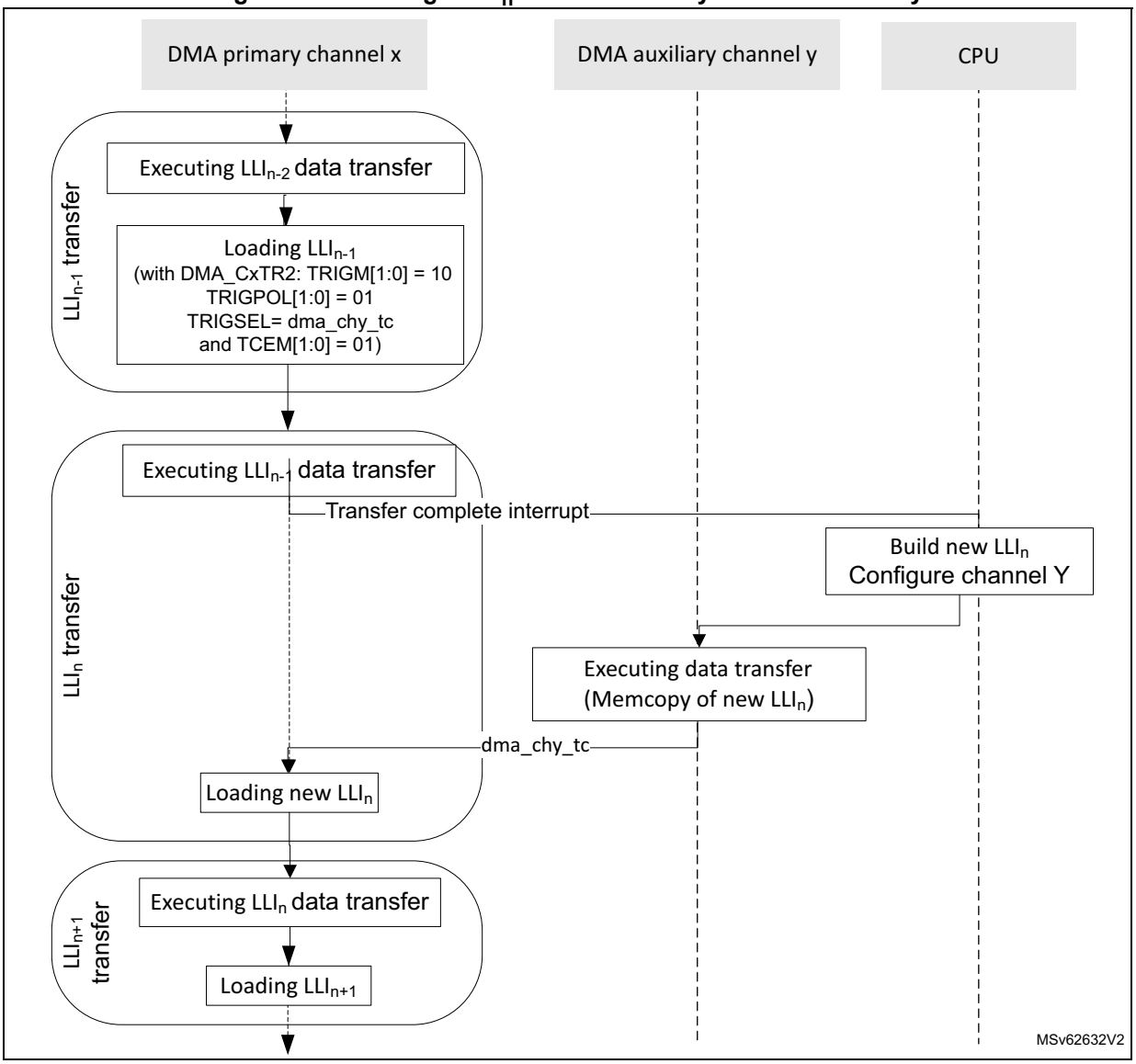
- TRIGM[1:0] = 10 (link transfer triggering mode)
- TRIGPOL[1:0] = 01 or 10 (rising or falling edge)
- TRIGSEL[4:0] (see *Section 16.3.5* for the trigger selection details)

Another auxiliary channel y can be used to store the channel x LLI_n in the memory and to generate a transfer complete event lpdma_chy_tc. By selecting this event as the input trigger of the link transfer of the LLI_{n-1} of the channel x, the software can pause the primary channel x after its LLI_{n-1} data transfer, until it is indeed written the LLI_n.

Figure 55 depicts such a dynamic elaboration of a linked-list of a primary channel x, via another auxiliary channel y.

Caution: This use case is restricted to an application with a LLI_{n-1} data transfer that does not need a trigger. The triggering mode of this LLI_{n-1} is used to load the next LLI_n.

Figure 55. Inserting a LLI_n with an auxiliary LPDMA channel y



16.4.8 LPDMA channel state and linked-list programming in link step mode

When LPDMA_CxCR.LSM = 1, a channel transfer is executed and completed after each single execution of a LLI, including its (conditional) data transfer and its (conditional) link transfer.

A LPDMA channel transfer can be programmed at LLI level, started by writing 1 into LPDMA_CxCR.EN, and after completed at LLI level:

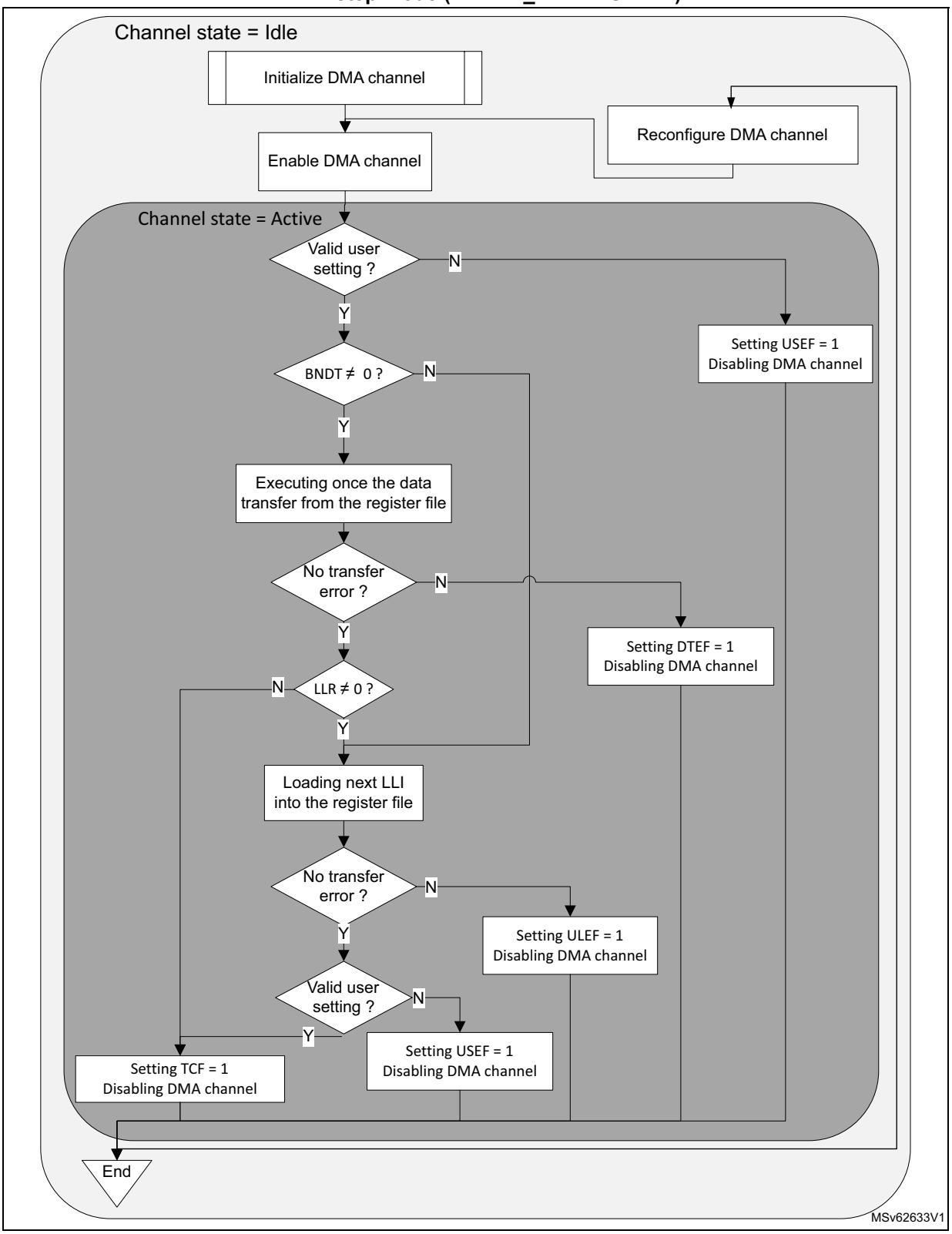
- The current LLI_n transfer is described with:
  - LPDMA_CxTR1 defines the source/destination elementary single transfers.
  - LPDMA_CxBR1 defines the number of bytes at a block level (BNDT[15:0]).
  - LPDMA_CxTR2 defines the input control (request, trigger) and the output control (transfer complete event) of the transfer.
  - LPDMA_CxSAR/LPDMA_CxDAR defines the source/destination transfer start address.
  - LPDMA_CxLLR defines the data structure and the address offset of the next LLI_{n+1} in the memory.
- The current LLI_n transfer is completed after the single execution of the current LLI_n:
  - after the (conditional) data transfer completion (when LPDMA_CxBR1.BNDT[15:0] = 0)
  - after the (conditional) update of the LPDMA link register file from the data structure of the next LLI_{n+1} in memory

Note: If a LLI is recursive (pointing to itself as a next LLI, either LPDMA_CxLLR.ULL = 1 and LPDMA_CxLLR.LA[15:2] is updated with the same value, or LPDMA_CxLLR.ULL = 0), a channel in link step mode is completed after each repeated single execution of this LLI.

Figure 56 depicts the LPDMA channel execution mode, and its programming in link step mode.

Note: Figure 56 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in LPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

Figure 56. LPDMA channel execution and linked-list programming in link step mode (LPDMA_CxCR.LSM = 1)

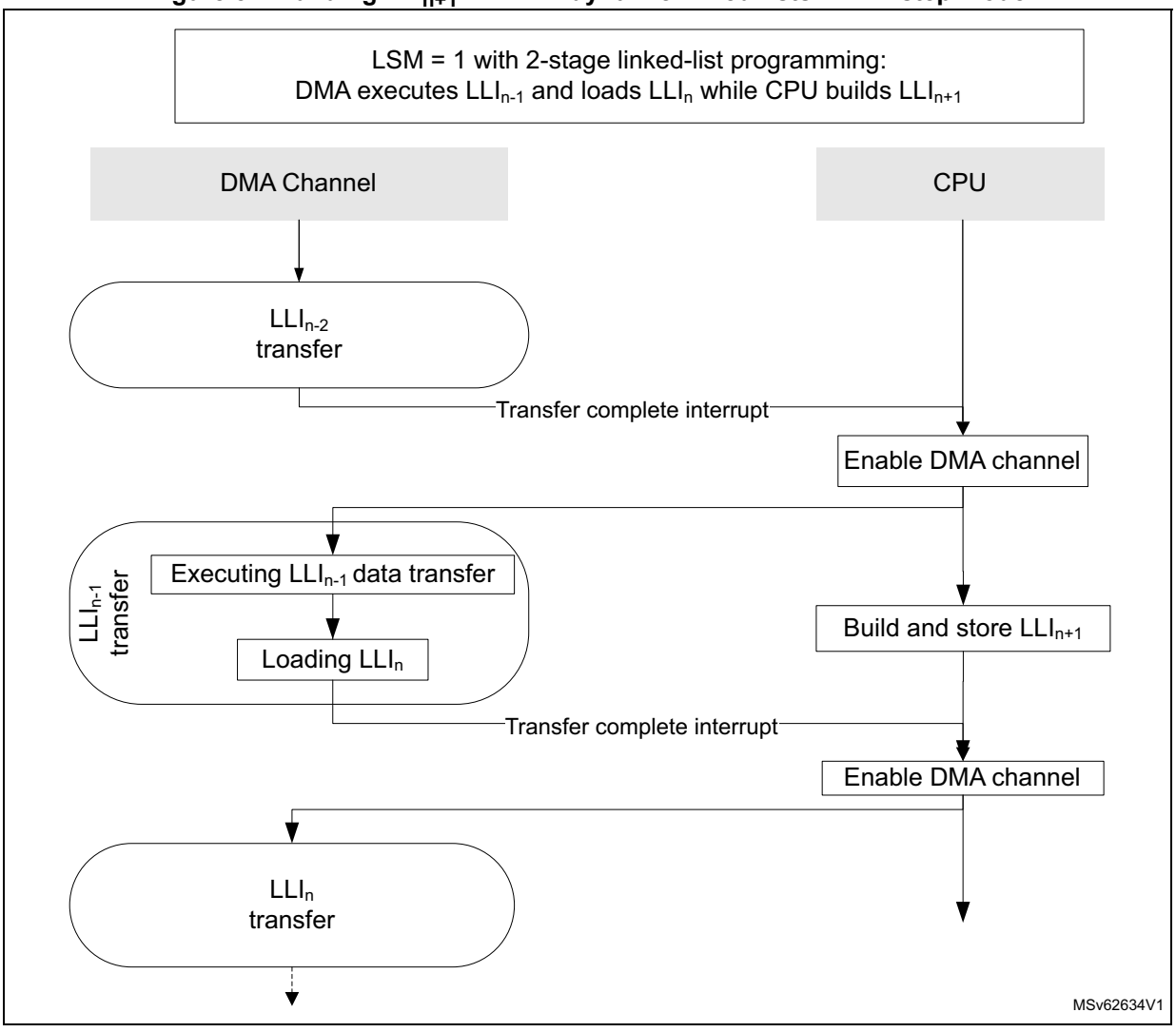


The link step mode can be used to elaborate dynamically LLIs in memory during run-time. The software can be facilitated by using a static data structure for any LLI_n (all update bits of LPDMA_CxLLR have a static value, LLI_n.LLR.LA = LLI_{n-1}.LLR.LA + constant).

Run-time adding a LLI_{n+1} in link step mode

During run-time, the software can defer the elaboration of the LLI_{n+1} (and next LLIs), until/after LPDMA executed the transfer from the LLI_{n-1} and loaded the LLI_n from the memory, as shown in the figure below.

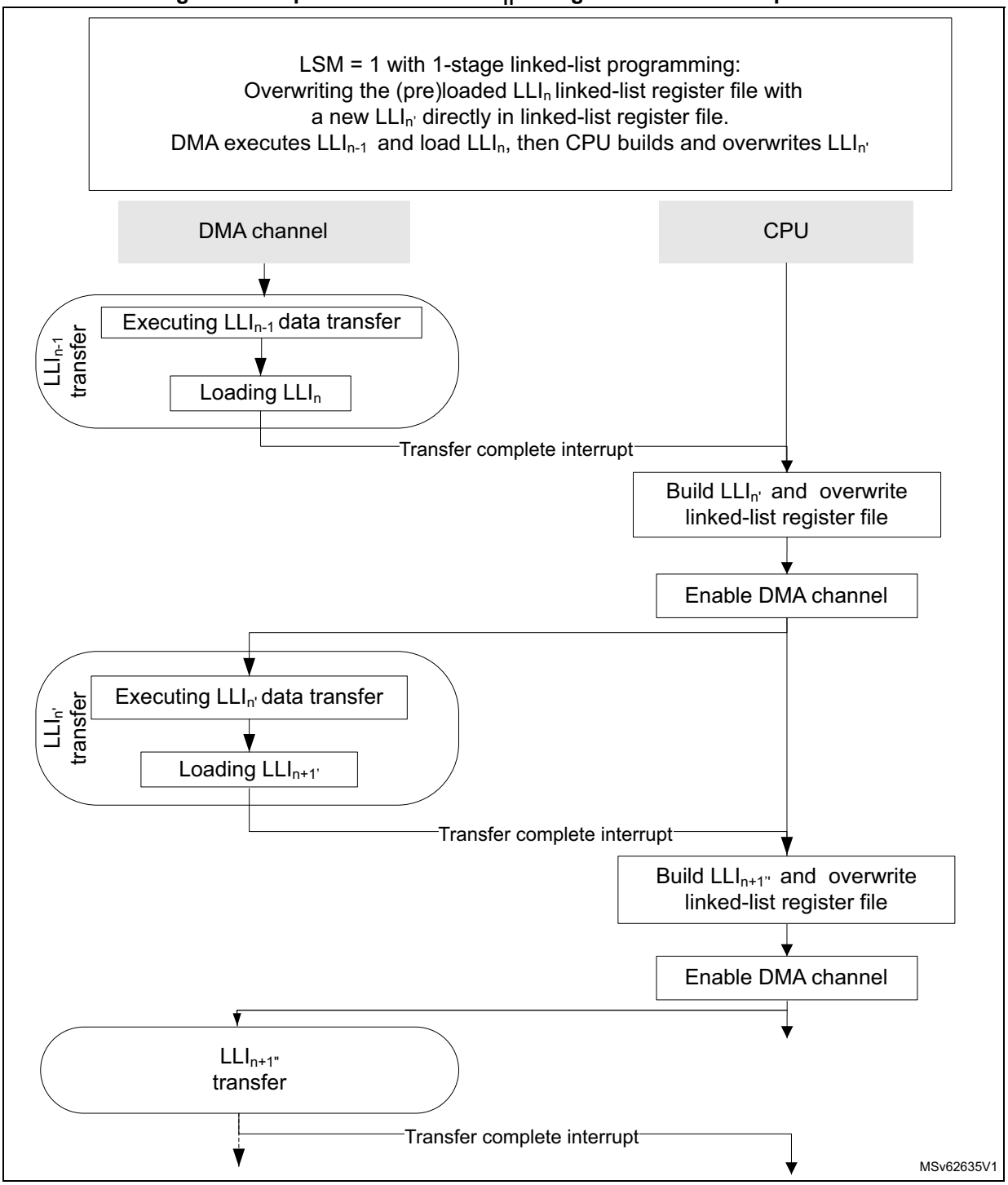
Figure 57. Building LLI_{n+1}: LPDMA dynamic linked-lists in link step mode



Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode (in linked-list register file)

In this link step mode, during run-time, the software can build and insert a new $LLI_{n'}$, after LPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory by overwriting directly the linked-list register file with the new $LLI_{n'}$, as shown in *Figure 58*.

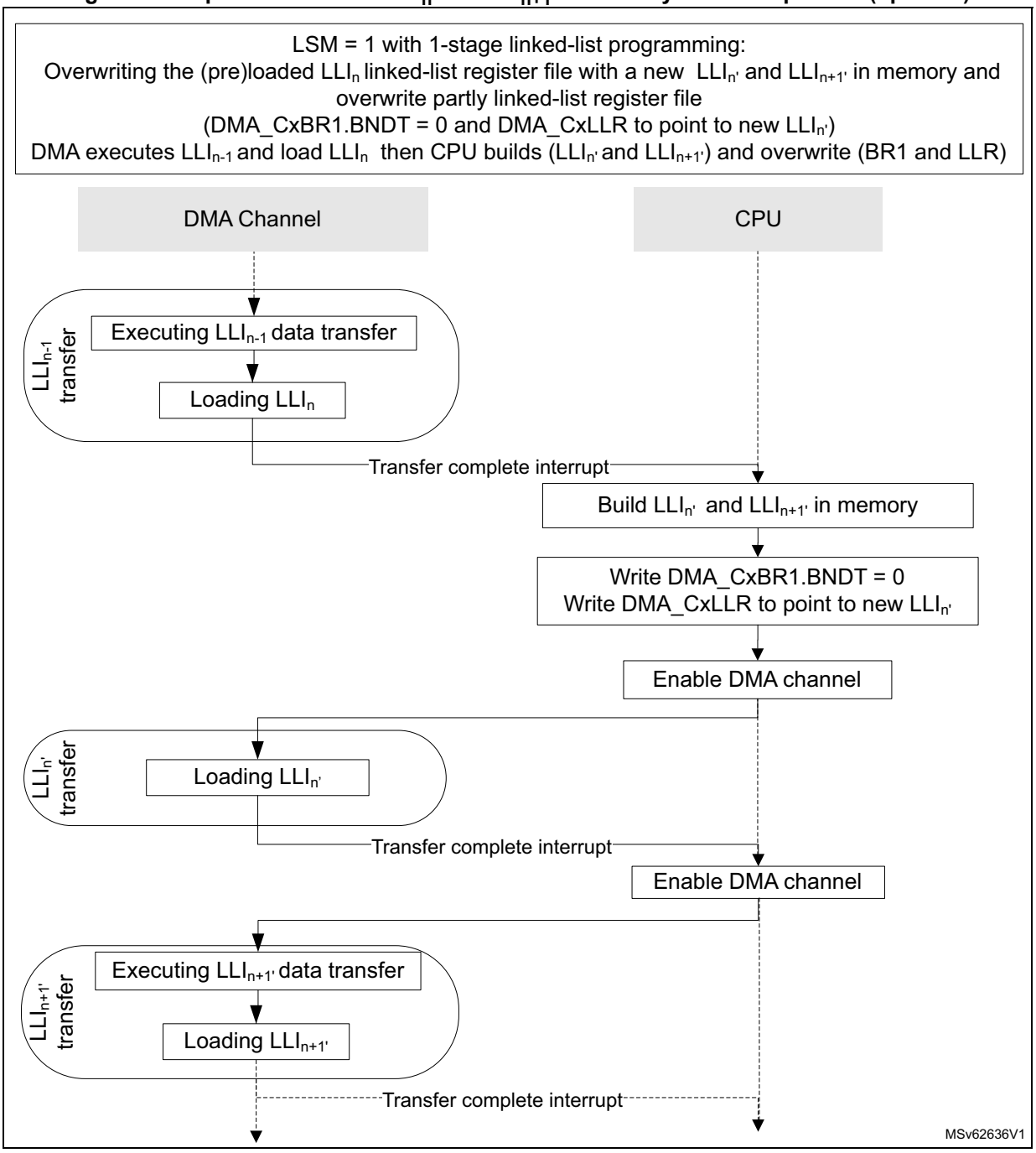
Figure 58. Replace with a new $LLI_{n'}$ in register file in link step mode



Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode (in the memory)

The software can build and insert a new $LLI_{n'}$ and $LLI_{n+1'}$ in the memory, after LPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory, by overwriting partly the linked-list register file (LPDMA_CxBR1.BNDT[15:0] to be null and LPDMA_CxLLR to point to new $LLI_{n'}$) as shown in *Figure 59*.

Figure 59. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 1)



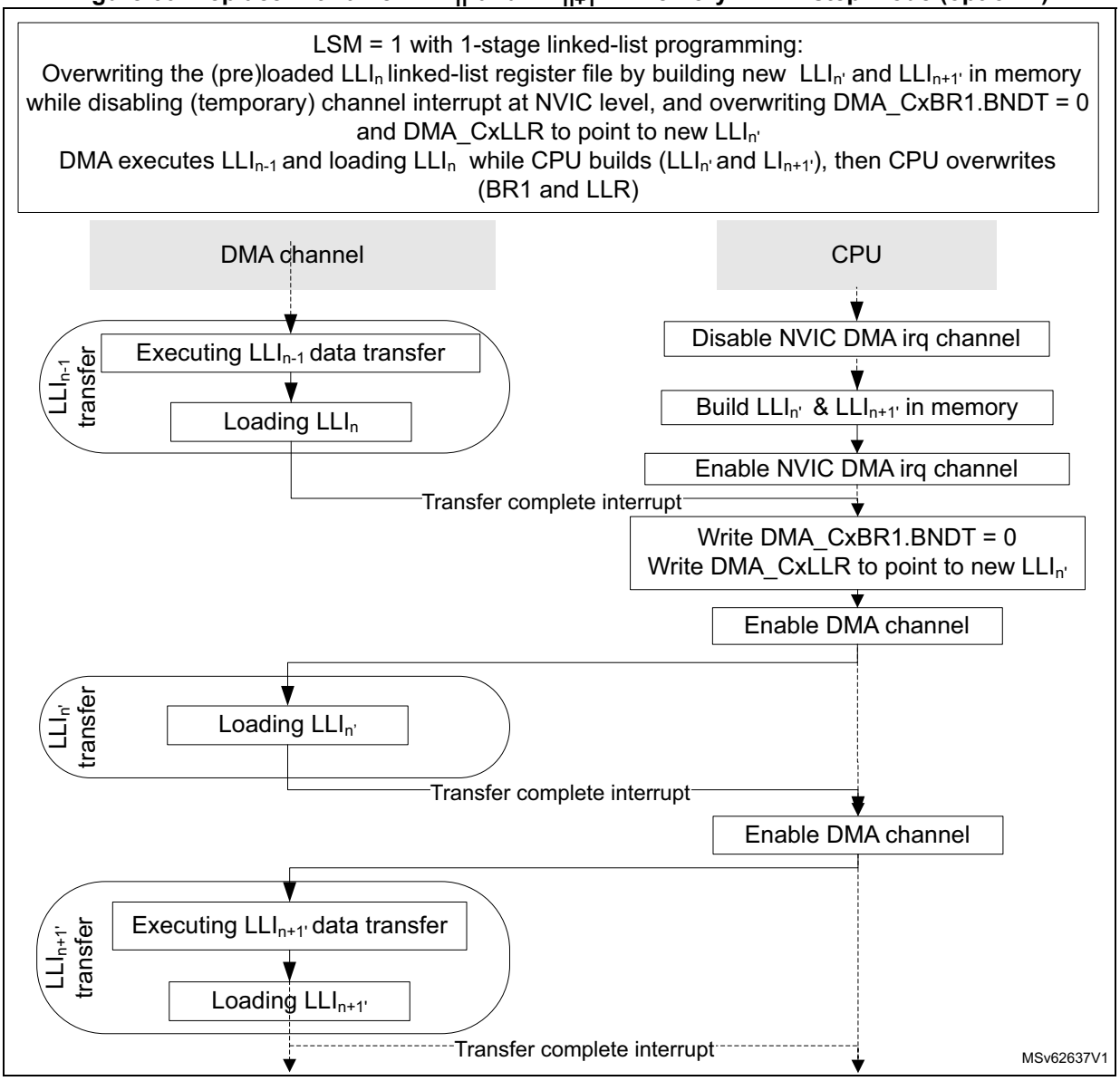
Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode

Other software implementations exist. Meanwhile LPDMA executes the transfer from the LLI_{n-1} and loads a formerly elaborated LLI_n from the memory (or even earlier), the software can do the following:

1. Disable the NVIC for not being interrupted by the interrupt handling.
2. Build a new $LLI_{n'}$ and a new $LLI_{n+1'}$.
3. Enable again the NVIC for the channel interrupt (transfer complete) notification.

The software in the interrupt handler for LLI_{n-1} is then restricted to overwrite LPDMA_CxBR1.BNDT[15:0] to be null and LPDMA_CxLLR to point to new $LLI_{n'}$, as shown in *Figure 60*.

Figure 60. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 2)



16.4.9 LPDMA channel state and linked-list programming

The software can reconfigure a channel when the channel is disabled (LPDMA_CxCR.EN = 0) and update the execution mode (LPDMA_CxCR.LSM) to change from/to run-to-completion mode to/from link step mode.

In any execution mode, the software can:

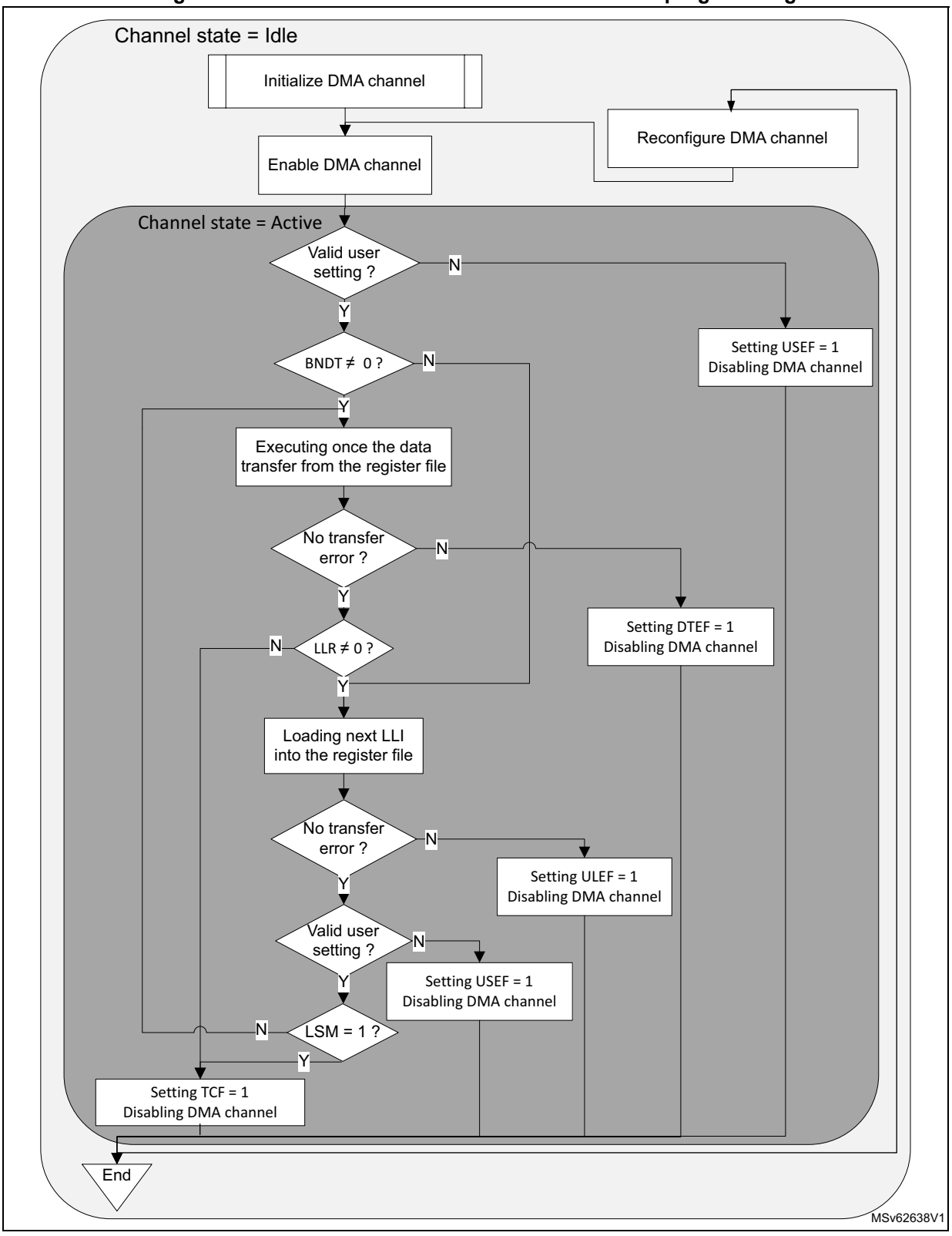
- reprogram LLI_{n+1} in the memory to finally complete the channel by this LLI_{n+1} (clear the LPDMA_CxLLR of this LLI_{n+1}), before this LLI_{n+1} is loaded/used by the LPDMA channel
- abort and reconfigure the channel with a LSM update (see *Section 16.4.4*)

In link step mode, the software can clear LSM after each a single execution of any LLI, during LLI_{n-1}.

Figure 61 shows the overall and unified LPDMA linked-list programming, whatever is the execution mode.

Note: Figure 61 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in LPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion the hardware always set TCEF = 1 and disables the channel. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

Figure 61. LPDMA channel execution and linked-list programming



16.4.10 LPDMA direct transfers

There is a single transfer operation mode called the direct mode. Any LPDMA channel is used in direct mode. Any channel is implemented without any FIFO (for every channel x, dma_fifo_size[x] = 0).

LPDMA single

A programmed transfer at the lowest level is called a LPDMA single.

A LPDMA single data width is 1, 2 or 4 bytes, as defined by the SDW_LOG2[1:0] and DDW_LOG2[1:0] fields of LPDMA_CxTR1(respectively for source and destination).

Note: The user must not assign a 8-byte data width (SDW_LOG2[1:0] = 0b11 or DDW_LOG2[1:0] = 0b11) else a user setting is reported and no transfer is issued.

The addressing mode after each data of a LPDMA single is defined by the SINC and DINC bits of LPDMA_CxTR1(respectively for source and destination): either a fixed addressing or an incremented addressing with contiguous data.

The start and next addresses of a LPDMA source/destination single (defined by LPDMA_CxSAR and LPDMA_CxDAR) must be aligned with the respective data width.

The table below lists the main characteristics of a LPDMA single.

The next single address in the table is the next source/destination address, pointed by LPDMA_CxSAR and LPDMA_CxDAR, once the programmed source/destination single is completed.

Table 135. Programmed LPDMA source/destination single

Programmed LPDMA source/destination single	SDW_LOG2[1:0] DDW_LOG2[1:0]	Data width (bytes)	SINC/ DINC	Addressing mode	SAR/DAR next single address	Address alignment
Fixed byte single	00	1	0	Fixed	+0	1
Fixed half-word single	01	2				2
Fixed word single	10	4				4
Incremented byte single	00	1	1	Incremented	+1	1
Incremented half-word single	01	2			+2	2
Incremented word single	10	4			+4	4
Forbidden	11	Causes USEF generation and none single to be issued.				

In direct mode, a LPDMA single is an AHB single transfer.

LPDMA data handling: byte-based padding/truncation, sign extension and left/right alignment

The user can configure some data manipulation between a transferred data from the source and its transfer to the destination. Data handling is controlled by the LPDMA_CxTR1 register:

- If destination data width = source data width (DDW_LOG2[1:0] = SDW_LOG2[1:0]), the source data is copied as is and transferred to the destination.
- Else, depending on PAM:
  - If destination data width > source data width, the source data can be either right-aligned and padded with 0 s, or sign extended up to the destination data width.
  - If destination data width < source data width, the source data can be either right-aligned and left-truncated down to the destination data width, or left-aligned and right-truncated down to the destination data width.

There is no data manipulation between two distinct transferred data from the source, before the generation of the destination transfer.

The table below lists possible data handling from the source to the destination.

Table 136. Programmed data handling

SDW_ LOG2[1:0]	Source data	Source data stream(1)	DDW_ LOG2[1:0]	Destination data	PAM[0]	Destination data stream(1)
00	Byte	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$	00	Byte	x	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
			01	Half-word	0 (RA, 0P)(2)(3)	$0B_3,0B_2,0B_1,0B_0$
					1 (RA, SE)(2)(4)	SB_3,SB_2,SB_1,SB_0
			10	Word	0 (RA, 0P)	$000B_1,000B_0$
					1 (RA, SE)	$SSSB_1,SSSB_0$
01	Half-word	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$	00	Byte	0 (RA, LT)(2)	B_6,B_4,B_2,B_0
					1 (LA, RT)(2)	B_7,B_5,B_3,B_1
			01	Half-word	xx	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
			10	Word	0 (RA, 0P)	$00B_3B_2,00B_1B_0$
					1 (RA, SE)	SSB_3B_2,SSB_1B_0
10	Word	$B_7B_6B_5B_4,B_3B_2B_1B_0$	00	Byte	0 (RA, LT)	B_{12},B_8,B_4,B_0
					1 (LA, RT)	B_{15},B_{11},B_7,B_3
			01	Half-word	0 (RA, LT)	B_5B_4,B_1B_0
				Word	1 (LA, RT)	B_7B_6,B_3B_2
			10		xx	$B_7B_6B_5B_4,B_3B_2B_1B_0$

1. Data stream is timely ordered starting from the byte with the lowest index a.k.a B_0.
2. RA = right aligned. LA = left aligned. RT= right truncated. LT = left truncated.
3. 0P = zero bit padding up to the destination data width.
4. SE = sign bit extended up to the destination data width.

16.4.11 LPDMA transfer request and arbitration

LPDMA transfer request

As defined by LPDMA_CxTR2, a programmed LPDMA data transfer is requested with one of the following:

- a software request if the control bit SWREQ = 1: This is used typically by the CPU for a data transfer from a memory-mapped address to another memory mapped address (memory-to-memory, GPIO to/from memory)
- an input hardware request coming from a peripheral if SWREQ = 0: The selection of the LPDMA hardware peripheral request is driven by the REQSEL[5:0] field (see *Section 16.3.4*). The selected hardware request can be one of the following:
  - a hardware request from a peripheral configured in LPDMA mode (for a transfer from/to the peripheral data register respectively to/from the memory)
  - a hardware request from a peripheral for its control registers update from the memory
  - a hardware request from a peripheral for a read of its status registers transferred to the memory

Caution: The user must not assign a same input hardware peripheral LPDMA request via LPDMA_CxTR.REQSEL[5:0] to two different channel if at a given time this request is asserted by the peripheral and each channel is ready to execute this requested data transfer. There is no user setting error reporting.

LPDMA transfer request for arbitration

For a given channel, a LPDMA requested data transfer from the source address to the destination address is arbitrated versus simultaneous requested LPDMA transfers from other channels, in order to be scheduled over the AHB master port. A LPDMA data transfer is atomic to the LPDMA arbitration: it consists of an AHB read single, immediately followed by an AHB write single. It is granted by the arbiter once for both AHB transfers, based on the channel priority defined by LPDMA_CxCR.PRIO[1:0].

An arbitrated LPDMA requested link transfer consists of one 32-bit read from the linked-list data structure in the memory to one of the linked-list registers (LPDMA_CxTR1, LPDMA_CxTR2, LPDMA_CxBR1, LPDMA_CxSAR, LPDMA_CxDAR or LPDMA_CxLLR). Each 32-bit read from the memory is arbitrated with the same channel priority as for data transfers, in order to be scheduled over the master port.

The re arbitration occurs after each granted single transfer:

- whatever how the requested data transfer is programmed:
  - with a software request for a memory-to-memory transfer (LPDMA_CxTR2.SWREQ = 1)
  - with a hardware request (LPDMA_CxTR2.SWREQ = 0) for a memory-to-peripheral transfer or a peripheral-to-memory transfer
- whatever the hardware request type

When the requested data transfer is programmed with a hardware request from a peripheral (LPDMA_CxTR2.SWREQ = 0), the first memory read of a block is gated by the occurrence of the corresponding and selected hardware request, whatever the peripheral is source or destination of the transfer. This first read request to the memory is not taken into account earlier by the arbiter (not as soon as the block transfer is enabled and executable).

LPDMA arbitration

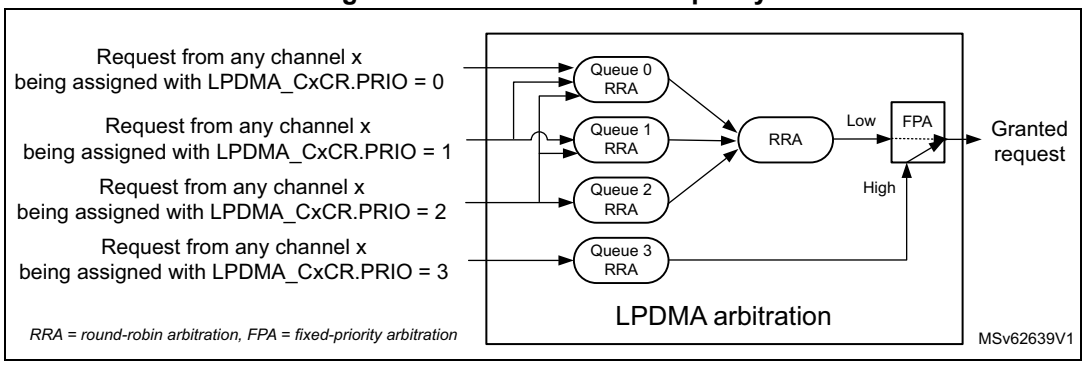
The LPDMA arbitration is directed from the 4-grade assigned channel priority (LPDMA_CxCR.PRIO[1:0]). The arbitration policy, as illustrated in *Figure 62*, is defined by:

- one high-priority traffic class (queue 3), dedicated to the assigned channels with priority 3, for time-sensitive channels

  This traffic class is granted via a fixed-priority arbitration against any other low-priority traffic class. Within this class, requested single transfers are round-robin arbitrated.
- three low-priority traffic classes (queues 0, 1 or 2) for non time-sensitive channels with priority 0, 1 or 2

  Each requested single transfer within this class is round-robin arbitrated, with a weight that is monotonically driven from the programmed priority:
  - Requests with priority 0 are allocated to the queue 0.
  - Requests with priority 1 are allocated and replicated to the queue 0 and queue 1.
  - Requests with priority 2 are allocated and replicated to the queue 0, queue 1, and queue 2.
  - Any queue 0, 1 or 2 equally grants any of its active input requests in a round-robin manner, provided there are simultaneous requests.
  - Additionally, there is a second stage for the low-traffic with a round-robin arbiter that fairly alternates between simultaneous selected requests from queue 0, queue 1 and queue 2.

Figure 62. LPDMA arbitration policy



1. RRA: round-robin arbitration
2. FPA: fixed-priority arbitration

LPDMA arbitration and bandwidth

With this arbitration policy, the following is guaranteed:

- Equal maximum bandwidth between requests with same priority
- Reserved bandwidth (noted as B_{Q3}) to the time-sensitive requests (with priority 3)
- Residual weighted bandwidth between different low-priority requests (priority 0 versus priority 1 versus priority 2).

The two following examples highlight that the weighted round-robin arbitration is driven by the programmed priorities:

- **Example 1**: basic application with two non time-sensitive LPDMA requests: req0 and req1. There are the following programming possibilities:
  - If they are assigned with same priority, the allocated bandwidth by the arbiter to req0 (B_{req0}) is **equal** to the allocated bandwidth to req1(B_{req1}).

    $B_{req0} = B_{req1} = 1/2 * (1 - B_{Q3})$
  - If req0 is assigned to priority 0 and req1 to priority 1, the allocated bandwidth to req0 (B_{P0}) is **3 times less** than the allocated bandwidth to req1 (B_{P1}).

    $B_{req0} = B_{P0} = 1/2 * 1/2 * (1 - B_{Q3}) = 1/4 * (1 - B_{Q3})$

    $B_{req1} = B_{P1} = (1/2 + 1) * 1/2 * (1 - B_{Q3}) = 3/4 * (1 - B_{Q3})$
  - If req0 is assigned to priority 0 and req1 to priority 2, the allocated bandwidth to req0 (B_{P0}) is **5 times less** than the allocated bandwidth to req1 (B_{P2}).

    $B_{req0} = B_{P0} = 1/2 * 1/3 * (1 - B_{Q3}) = 1/6 * (1 - B_{Q3})$

    $B_{req1} = B_{P2} = (1/2 + 1 + 1) * 1/3 * (1 - B_{Q3}) = 5/6 * (1 - B_{Q3})$

  The above computed bandwidth calculation is based on a theoretical input request, always active for any LPDMA clock cycle. This computed bandwidth from the arbiter must be weighted by the frequency of the request given by the application, that cannot be always active and may be quite much variable from one LPDMA client (example I2C at 400 kHz) to another one (PWM at 1 kHz) than the above x3 and x5 ratios.
- **Example 2**: application where the user distributes a same non-null N number of LPDMA requests to every non time-sensitive priority 0, 1 and 2. The bandwidth calculation is then the following:
  - The allocated bandwidth to the set of requests of priority 0 (B_{P0}) is

    $B_{P0} = 1/3 * 1/3 * (1 - B_{Q3}) = 1/9 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 1(B_{P1}) is

    $B_{P1} = (1/3 + 1/2) * 1/3 * (1 - B_{Q3}) = 5/18 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 2(B_{P2}) is

    $B_{P2} = (1/3 + 1/2 + 1) * 1/3 * (1 - B_{Q3}) = 11/18 * (1 - B_{Q3})$
  - The allocated bandwidth to any request n ($B_{n)}$ among the N requests of that priority Pi (i = 0 to 2) is $B_n = 1/N * B_{Pi}$
  - The allocated bandwidth to any request n of priority 0i ($B_{n, Pi}$) is

    $B_{n, P0} = 1/N * 1/9 * (1 - B_{Q3})$

    $B_{n, P1} = 1/N * 5/18 * (1 - B_{Q3})$

    $B_{n, P2} = 1/N * 11/18 * (1 - B_{Q3})$

  In this example, when the master port bus bandwidth is not totally consumed by the time-sensitive queue 3, the residual bandwidth is such that 2.5 times less bandwidth is allocated to any request of priority 0 versus priority 1, and 5.5 times less bandwidth is allocated to any request of priority 0 versus priority 2.

More generally, assume that the following requests are present:

- I requests (I ? 0) assigned to priority 0
  If I > 0, these requests are noted from i = 0 to I-1.
- J requests (J ? 0) assigned to priority 1
  If J > 0, these requests are noted from j = 0 to J-1.
- K requests (K > 0) assigned to priority 2
  These requests are noted from k = 0 to K-1
- L requests (L ? 0) assigned to priority 3
  If L > 0, these requests are noted from l = 0 to L-1.

As B_{Q3} is the reserved bandwidth to time-sensitive requests, the bandwidth for each request L with priority 3 is:

- $B_l = B_{Q3} / L$ for L > 0 (else: $B_l = 0$)

The bandwidth for each non-time sensitive queue is:

- $B_{Q0} = 1/3 * (1 - B_{Q3})$
- $B_{Q1} = 1/3 * (1 - B_{Q3})$
- $B_{Q2} = 1/3 * (1 - B_{Q3})$

The bandwidth for the set of requests with priority 0 is:

- $B_{P0} = I / (I + J + K) * B_{Q0}$

The bandwidth for each request i with priority 0 is:

- $B_i = B_{P0} / I$ for L > 0 (else $B_{P0} = 0$)

The bandwidth for the set of requests with priority 1 and routed to queue 0 is:

- $B_{P1,Q0} = J / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 1 and routed to queue 1 is:

- $B_{P1,Q1} = J / (J + K) * B_{Q1}$

The total bandwidth for the set of requests with priority 1 is:

- $B_{P1} = B_{P1,Q0} + B_{P1,Q1}$

The bandwidth for each request j with priority 1 is:

- $B_j = B_{P1} / J$ for J > 0 (else $B_j = 0$)

The bandwidth for the set of requests with priority 2 and routed to queue 0 is:

- $B_{P2,Q0} = K / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 2 and routed to queue 1 is:

- $B_{P2,Q1} = K / (J + K) * B_{Q1}$

The bandwidth for the set of requests with priority 2 and routed to queue 2 is:

- $B_{P2,Q2} = B_{Q2}$

The total bandwidth for the set of requests with priority 2 is:

- $B_{P2} = B_{P2,Q0} + B_{P2,Q1} + B_{P2,Q2}$

The bandwidth for each request k with priority 2 is:

- $B_k = B_{P2} / K$ (K>0 in the general case)

Thus finally the maximum allocated residual bandwidths for any i, j, k non-time sensitive request are:

- in the general case (when there is at least one request k with a priority 2 (K > 0)):
  - $B_i = 1/I * 1/3 * I/(I + J + K) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/3 *[J/(I + J + K) + J/(J + K)] * (1 - B_{Q3})$
  - $B_k = 1/K * 1/3 *[K/(I + J + K) + K/(J + K) + 1] * (1 - B_{Q3})$
- in the specific case (when there is no request k with a priority 2 (K = 0)):
  - $B_i = 1/I * 1/2 * I/(I + J) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/2 *[J/(I + J) + 1] * (1 - B_{Q3})$

Consequently, the LPDMA arbiter can be used as a programmable weighted bandwidth limiter, for each queue and more generally for each request/channel. The different weights are monotonically resulting from the programmed channel priorities.

16.4.12 LPDMA triggered transfer

A programmed LPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event, as defined by LPDMA_CxTR2.TRIGPOL[1:0] and LPDMA_CxTR2.TRIGSEL[4:0] (see *Section 16.3.5* for the trigger selection).

The triggered transfer, as defined by the trigger mode in LPDMA_CxTR2.TRIGM[1:0], can be at LLI data transfer level, to condition the first single read of a block, or each programmed single read. The trigger mode can also be programmed to condition the LLI link transfer (see the TRIGM[1:0] description in *LPDMA channel x transfer register 2 (LPDMA_CxTR2)* for more details).

Trigger hit memorization and trigger overrun flag generation

The LPDMA monitoring of a trigger for a channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (respectively TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

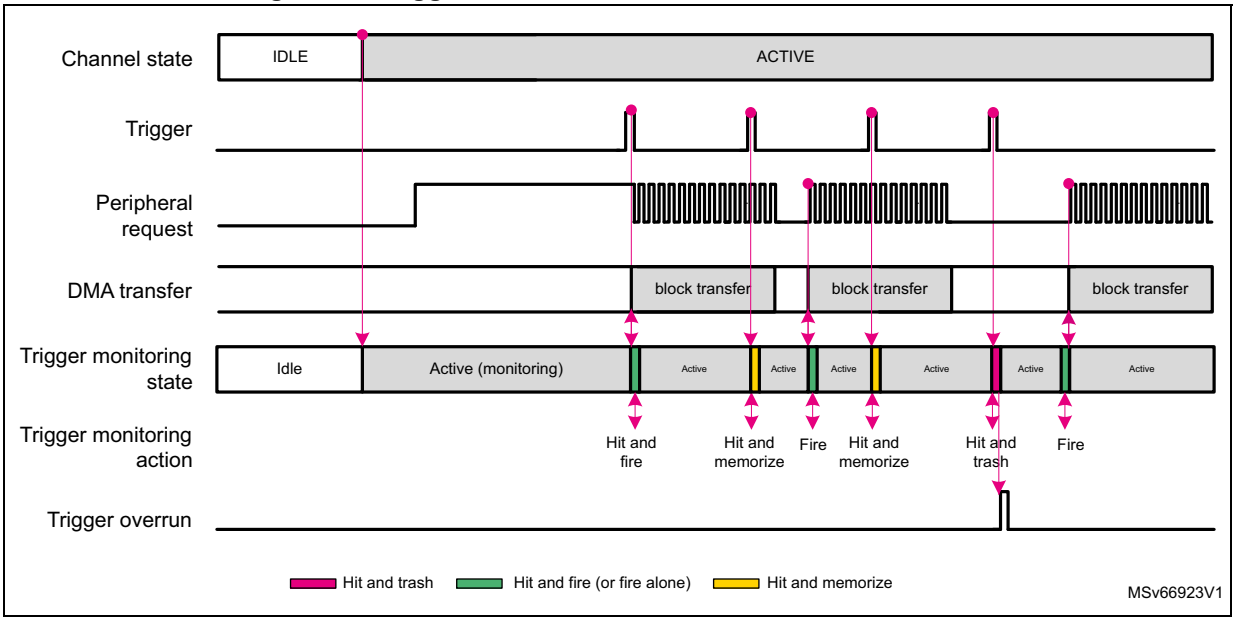
The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer. If a new trigger is detected, this hit is internally memorized to grant the next transfer, as long as the defined rising/falling edge and TRIGSEL[4:0] are not modified, and the channel is enabled.

Transferring a next $LLI_{n+1,}$ that updates the LPDMA_CxTR2 with a new value for any of TRIGSEL[4:0] or TRIGPOL[1:0], resets the monitoring, trashing the possible memorized hit of the formerly defined LLI_n trigger.

Caution: After a first new trigger hit_{n+1} is memorized, if another trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (LPDMA_CxSR.TOF = 1) and an interrupt is generated if enabled (if LPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

The figure below illustrates the trigger hit, memorization and overrun in the configuration example with a block-level trigger mode and a rising edge trigger polarity.

Figure 63. Trigger hit, memorization and overrun waveform



Note: The user can assign the same input trigger event to different channels. This can be used to trigger different channels on a broadcast trigger event.

16.4.13 LPDMA circular buffering with linked-list programming

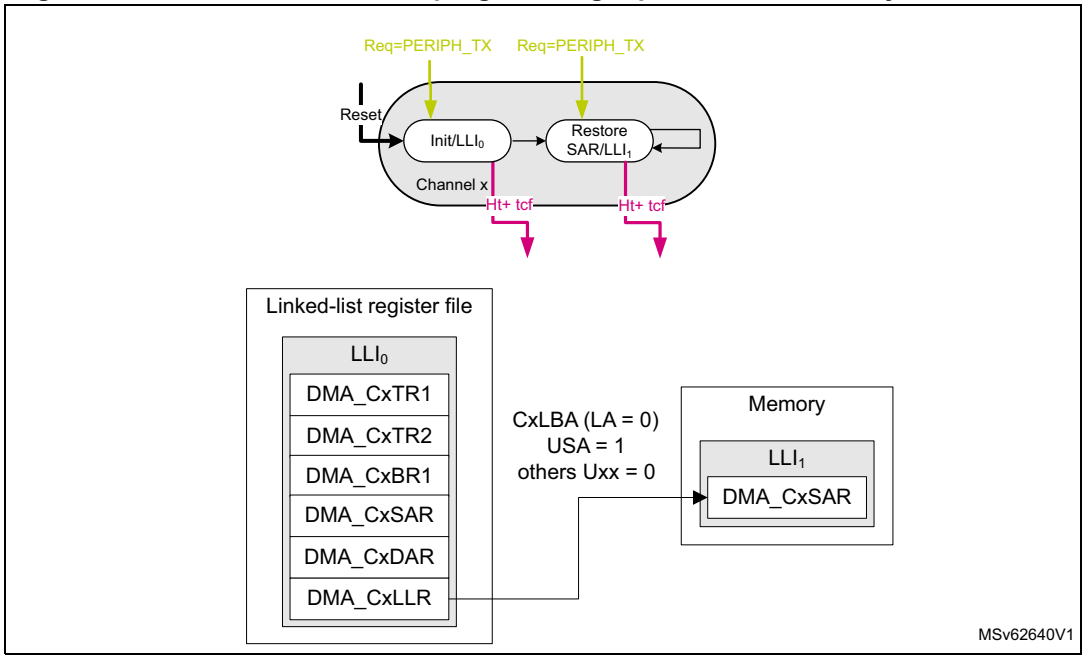
LPDMA circular buffering for memory-to-peripheral and peripheral-to-memory transfers

For a circular buffering, with a continuous memory-to-peripheral (or peripheral-to-memory) transfer, the software must set up a channel with half-transfer and complete transfer events/interrupts generation (LPDMA_CxCR.HTIE = 1 and LPDMA_CxCR.TCIE = 1), in order to enable a concurrent buffer software processing.

LLI_0 is configured for the first block transfer. A continuously-executed LLI_1 is needed to restore the memory source (or destination) start address, for the memory-to-peripheral transfer (respectively the peripheral-to-memory transfer). LPDMA automatically reloads the initially programmed LPDMA_CxBR1.BNDT[15:0] when a block transfer is completed, and there is no need to restore LPDMA_CxBR1.

The figure below illustrates this programming with a LPDMA channel and a source circular buffer.

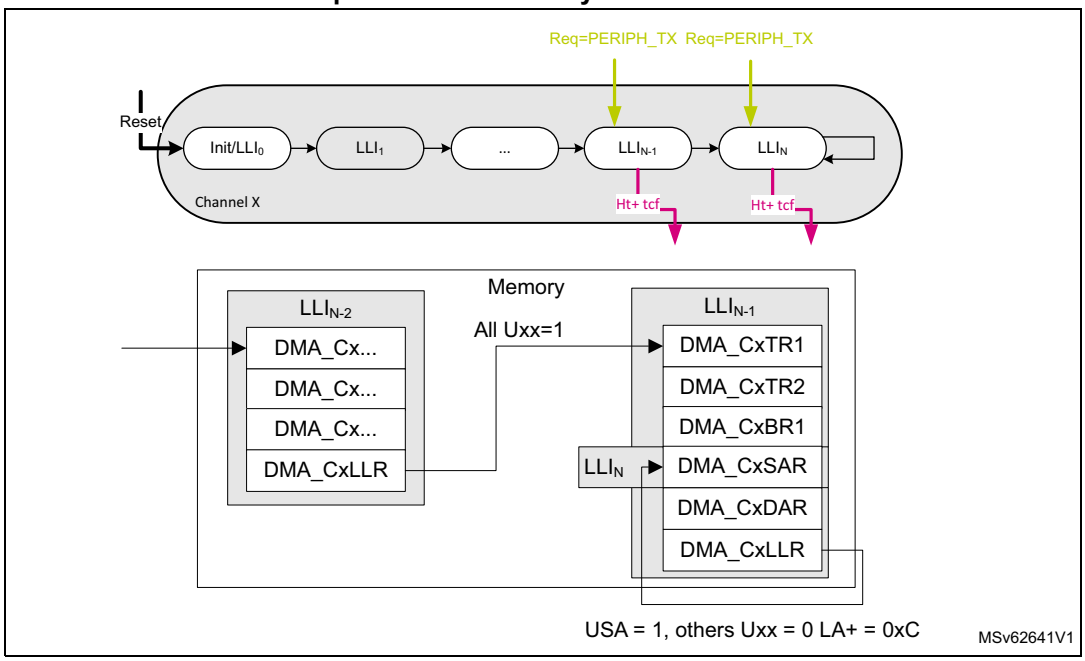
Figure 64. LPDMA circular buffer programming: update of the memory start address



If circular buffering must be executed after some other transfers over the shared LPDMA channel x, the before-last LLI_{N-1} in memory is needed to configure the first block transfer. And the last LLI_N restores the memory source (or destination) start address in memory-to-peripheral transfer (respectively in peripheral-to-memory transfer).

The figure below illustrates this programming with a shared LPDMA channel, and a source circular buffer.

Figure 65. Shared LPDMA channel with circular buffering: update of the memory start address



16.4.14 LPDMA secure/nonsecure channel

The LPDMA controller is compliant with the TrustZone hardware architecture at channel level, partitioning all its resources so that they exist in one of the secure and nonsecure worlds at any given time.

Any channel x is a secure or a nonsecure hardware resource, as configured by LPDMA_SECCFGR.SECx.

When a channel x is configured in secure state by a secure and privileged agent, the following access control rules are applied:

- A nonsecure read access to a register field of this channel is forced to return 0, except for LPDMA_SECCFGR, LPDMA_PRIVCFGR, LPDMA_RCFGLOCKR that are readable by a nonsecure agent.
- A nonsecure write access to a register field of this channel has no impact.

When a channel x is configured in secure state, a secure agent can configure separately as secure or nonsecure the LPDMA data transfer from the source (LPDMA_CxTR1.SSEC) and the LPDMA data transfer to the destination (LPDMA_CxTR1.DSEC).

When a channel x is configured in secure state and in linked-list mode, the loading of the next linked-list data structure from the LPDMA memory into its register file, is automatically performed with secure transfers via the master port.

LPDMA generates the DMA channel state versus security, reflecting LPDMA_SECCFGR, to keep the other peripherals informed of the secure/nonsecure state of each LPDMA channel x.

LPDMA also generates a security illegal access pulse signal on an illegal nonsecure access to a secure LPDMA register. This signal is routed to the TrustZone interrupt controller.

When the secure software must switch a channel from a secure state to a nonsecure state, the secure software must abort the channel or wait until the secure channel is completed before switching. This is needed to dynamically re-allocate a channel to a next nonsecure transfer as a nonsecure software is not allowed to do so and must have LPDMA_CxCR.EN = 0 before the nonsecure software can reprogram the LPDMA_CxCR for a next transfer. The secure software may reset not only the channel x (LPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

16.4.15 LPDMA privileged/unprivileged channel

Any channel x is a privileged or unprivileged hardware resource, as configured by a privileged agent via LPDMA_PRIVCFGR.PRIVx.

When a channel x is configured in a privileged state by a privileged agent, the following access control rules are applied:

- An unprivileged read access to a register field of this channel is forced to return 0, except for LPDMA_PRIVCFGR, LPDMA_SECCFGR, that are readable by an unprivileged agent.
- An unprivileged write access to a register field of this channel has no impact.

When a channel is configured in a privileged (or unprivileged) state, the source and destination data transfers are privileged (respectively unprivileged) transfers over the AHB master port.

When a channel is configured in a privileged (or unprivileged) state and in linked-list mode, the loading of the next linked-list data structure from the LPDMA memory into its register file, is automatically performed with privileged (respectively unprivileged) transfers, via the master port.

LPDMA generates a DMA channel state versus privilege, reflecting LPDMA_PRIVCFGR, to keep the other peripherals informed of the privileged/unprivileged state of each DMA channel x.

Additionally, the LPDMA generates the privileged illegal access pulse signal on an illegal unprivileged access to a privileged LPDMA register. This signal may be used or not, depending on the product (see the system security section for more details).

When the privileged software must switch a channel from a privileged state to an unprivileged state, the privileged software must abort the channel or wait until that the privileged channel is completed before switching. This is needed to dynamically re-allocate a channel to a next unprivileged transfer as an unprivileged software is not allowed to do so, and must have LPDMA_CxCR.EN = 0 before the unprivileged software can reprogram the LPDMA_CxCR for a next transfer. The privileged software may reset not only the channel x (LPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

16.4.16 LPDMA error management

LPDMA is able to manage and report to the user a transfer error, as follows, depending on the root cause.

Data transfer error

On a bus access (as a AHB single) to the source or the destination:

- The source or destination target reports an AHB error.
- The programmed channel transfer is stopped (LPDMA_CxCR.EN cleared by the LPDMA hardware). The channel status register reports an idle state (LPDMA_CxSR.IDLEF = 1) and the data error (LPDMA_CxSR.DTEF = 1).
- After a LPDMA data transfer error, the user must perform a debug session, taking care of the product-defined memory mapping of the source and destination, including the protection attributes.
- After a LPDMA data transfer error, the user must issue a channel reset (set LPDMA_CxCR.RESET) to reset the hardware LPDMA channel data path before the user enables again the same channel for a next transfer.

Link transfer error

On a tentative update of a LPDMA channel register from the programmed LLI in the memory:

- The linked-list memory reports an AHB error.
- The programmed channel transfer is stopped (LPDMA_CxCR.EN cleared by the LPDMA hardware), the channel status register reports an idle state (LPDMA_CxSR.IDLEF = 1) and the link error (LPDMA_CxSR.ULEF = 1).
- After a LPDMA link error, the user must perform a debug session, taking care of the product-defined memory mapping of the linked-list data structure (LPDMA_CxLBAR and LPDMA_CxLLR), including the protection attributes.
- After a LPDMA link error, the user must explicitly write the linked-list register file (LPDMA_CxTR1, LPDMA_CxTR2, LPDMA_CxBR1, LPDMA_CxSAR, LPDMA_CxDAR and LPDMA_CxLLR), before the user enables again the same channel for a next transfer.

User setting error

On a tentative execution of a LPDMA transfer with an unauthorized user setting:

- The programmed channel transfer is disabled (LPDMA_CxCR.EN forced and cleared by the LPDMA hardware) preventing the next unauthorized programmed data transfer from being executed. The channel status register reports an idle state (LPDMA_CxSR.IDLEF = 1) and a user setting error (LPDMA_CxSR.USEF = 1).
- After a LPDMA user setting error, the user must perform a debug session, taking care of the LPDMA channel programming. A user setting error can be caused by one of the following:
  - a programmed null source block size without a programmed update of this value from the next LLI_1 (LPDMA_CxBR1.BNDT[15:0] = 0 and LPDMA_CxLLR.UB1 = 0)
  - a programmed non-null source block size being not a multiple of the programmed data width of a source single transfer (LPDMA_CxBR1.BNDT[2:0] versus LPDMA_CxTR1.SDW_LOG2[1:0])

- a programmed unaligned source start address, being not a multiple of the programmed data width of a source single transfer (LPDMA_CxSAR[2:0] versus LPDMA_CxTR1.SDW_LOG2[1:0])
- a programmed unaligned destination start address, being not a multiple of the programmed data width of a destination single transfer (LPDMA_CxDAR[2:0] versus LPDMA_CxTR1.DDW_LOG2[1:0])
- a programmed double-word source data width (LPDMA_CxTR1.SDW_LOG2[1:0] = 0b11)
- a programmed double-word destination data width (LPDMA_CxTR1.DDW_LOG2[1:0] = 0b11
- a programmed linked-list item LLI_{n+1} with a null data transfer (LPDMA_CxLLR.UB1 = 1 and LPDMA_CxBR1. BNDT = 0)

16.4.17 LPDMA autonomous mode

To save dynamic power consumption while LPDMA executes the programmed linked-list transfers, LPDMA hardware automatically manages its own clock gating and generates a clock request output signal to the RCC, whenever the device is in Run, Sleep or Stop mode, provided that the RCC is programmed with the corresponding LPDMA enable control bits.

For more details about the RCC programming, refer to the RCC section of the reference manual.

For mode details about the availability of the LPDMA autonomous feature vs the device low-power modes, refer to *Section 16.3.2*.

Note: design/firmware: In low power modes, DMA clock request is asserted upon a DMA request/trigger from an autonomous peripheral or upon a DMA trigger from an EXTI line.

The user can program and schedule the execution of a given LPDMA transfer at a LLI_n level of a LPDMA channel x, with LPDMA_CxTR2 as follows:

- The software controls and conditions the input of a transfer with TRIGM[1:0], TRIGPOL[1:0], TRIGSEL[y[(a)]:0], SWREQ, and REQSEL[z[(a)]:0] for the input trigger and request.
- The software controls and signals the output of a transfer with TCEM[1:0] for generating or not a transfer complete event, and generating or not an associated half-transfer event).

See *LPDMA channel x transfer register 2 (LPDMA_CxTR2)* for more details.

The output channel x transfer complete event, lpdma_chx_tc, can be programmed as a selected input trigger for a channel if this event is looped-back and connected at the LPDMA level (see *Section 16.3.5*), allowing autonomous and fine DMA inter-channel transfer scheduling, without needing a cleared transfer complete flag (TCF).

a. Refer to *LPDMA channel x transfer register 2 (LPDMA_CxTR2)* for y and z values.

A given LPDMA channel x asserts its clock request in one of the following conditions:

- if the next transfer to be executed is programmed as conditioned by a trigger (LPDMA_CxTR2.TRIGPOL[1:0] and LPDMA_CxTR2.TRIGM[1:0]), only when the trigger hit occurs.
- if the next transfer to be executed is not conditioned by a trigger:
  - if LPDMA_CxTR2.SWREQ = 0, only when the hardware request is asserted by the selected peripheral
  - if LPDMA_CxTR2.SWREQ = 1 (memory-to-memory, GPIO to/from memory), as soon as the LPDMA is enabled

The LPDMA channel x releases its clock request as soon as all the following conditions are met:

- The transfer to be executed is completed.
- The LPDMA channel x is not immediately ready and requested to execute the next transfer.
- If a channel x interrupt was raised, all the flags of the status register that can cause this interrupt, are cleared by a software agent.

When one channel asserts its clock request, the LPDMA asserts its clock request to the RCC. When none channel asserts its clock request, the LPDMA releases its clock request to the RCC.

16.5 LPDMA in debug mode

When the microcontroller enters debug mode (core halted), any channel x can be individually either continued (default) or suspended, depending on the programmable control bit in the DBGMCU module.

Note: In debug mode, LPDMA_CxSR.SUSPF is not altered by a suspension from the programmable control bit in the DBGMCU module. In this case, LPDMA_CxSR.IDLEF can be checked to know the completion status of the channel suspension.

16.6 LPDMA in low-power modes

Table 137. Effect of low-power modes on LPDMA

Mode	Description
Sleep	No effect. LPDMA interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the LPDMA registers is kept when entering Stop mode. The content of the LPDMA registers can be autonomously updated by a next linked-list item from memory, to perform autonomous data transfers. LPDMA interrupts can cause the device to exit Stop mode.
Standby	The LPDMA is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 16.3.2* to know which Stop mode is supported.

16.7 LPDMA interrupts

There is one LPDMA interrupt line for each channel, and separately for each CPU (if several ones in the devices).

Table 138. LPDMA interrupt requests

Interrupt acronym	Interrupt event	Interrupt enable	Event flag	Event clear method
LPDMA_CHx	Transfer complete	LPDMA_CxCR.TCIE	LPDMA_CxSR.TCF	Write 1 to LPDMA_CxFCR.TCF
	Half transfer	LPDMA_CxCR.HTIE	LPDMA_CxSR.HTF	Write 1 to LPDMA_CxFCR.HTF
	Data transfer error	LPDMA_CxCR.DTEIE	LPDMA_CxSR.DTEF	Write 1 to LPDMA_CxFCR.DTEF
	Update link error	LPDMA_CxCR.ULEIE	LPDMA_CxSR.ULEF	Write 1 to LPDMA_CxFCR.ULEF
	User setting error	LPDMA_CxCR.USEIE	LPDMA_CxSR.USEF	Write 1 to LPDMA_CxFCR.USEF
	Suspended	LPDMA_CxCR.SUSPIE	LPDMA_CxSR.SUSPF	Write 1 to LPDMA_CxFCR.SUSPF
	Trigger overrun	LPDMA_CxCR.TOFIE	LPDMA_CxSR.TOF	Write 1 to LPDMA_CxFCR.TOF

A LPDMA channel x event may be:

- a transfer complete
- a half-transfer complete
- a transfer error, due to either:
  - a data transfer error
  - an update link error
  - a user setting error completed suspension
- a trigger overrun

Note: When a channel x transfer complete event occurs, the output signal lpdma_chx_tc is generated as a high pulse of one clock cycle.

An interrupt is generated following any xx event, provided that both:

- the corresponding interrupt event xx is enabled (LPDMA_CxCR.xxIE = 1)
- the corresponding event flag is cleared (LPDMA_CxSR.xxF = 0). This means that, after a previous same xx event occurrence, a software agent must have written 1 into the corresponding xx flag clear control bit (write 1 into LPDMA_CxFCR.xxF).

TCF (transfer complete) and HTF (half transfer) events generation is controlled by LPDMA_CxTR2.TCEM[1:0] as follows:

- A transfer complete event is a block transfer complete or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode LPDMA_CxTR2.TCEM[1:0].
- A half-transfer event is a half-block transfer. A half-block transfer occurs when half of the source block size bytes (rounded-up integer of LPDMA_CxBR1.BNDT[15:0] / 2) is transferred to the destination.

See *LPDMA channel x transfer register 2 (LPDMA_CxTR2)* for more details.

A transfer error rises in one of the following situations:

- during a single data transfer from the source or to the destination (DTEF)
- during an update of a LPDMA channel register from the programmed LLI in memory (ULEF)
- during a tentative execution of a LPDMA channel with an unauthorized setting (USEF)

  The user must perform a debug session to correct the LPDMA channel programming versus the USEF root causes list (see *Section 16.4.16*).

A trigger overrun is described in *Trigger hit memorization and trigger overrun flag generation*.

16.8 LPDMA registers

The LPDMA registers must be accessed with an aligned 32-bit word data access.

16.8.1 LPDMA secure configuration register (LPDMA_SECCFGR)

Address offset: 0x000

Reset value: 0x0000 0000

A write access is ignored at bit level if the corresponding channel x is locked (LPDMA_RCFGLOCKR.LOCKx = 1).

A write access to this register must be secure and privileged. A read access is secure or nonsecure, privileged or unprivileged.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of a LPDMA channel (when LPDMA_CxCR.EN = 0), to securely allocate individually any channel x to the secure or nonsecure world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SECx**: secure state of channel x (x = 7 to 0)

0: nonsecure

1: secure

16.8.2 LPDMA privileged configuration register (LPDMA_PRIVCFGR)

Address offset: 0x004

Reset value: 0x0000 0000

A write access to this register must be privileged. A read access can be privileged or unprivileged, secure or nonsecure.

A write access is ignored at bit level if the corresponding channel x is locked (LPDMA_RCFGLOCKR.LOCKx = 1).

This register can mix secure and nonsecure information. If a channel x is configured as secure (LPDMA_SECCFGR.SECx = 1), the PRIVx bit can be written only by a secure (and privileged) agent.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of a LPDMA channel (LPDMA_CxCR.EN = 0), to individually allocate any channel x to the privileged or unprivileged world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRIVx**: privileged state of channel x (x = 7 to 0)
0: unprivileged
1: privileged

16.8.3 LPDMA configuration lock register (LPDMA_RCFGLOCKR)

Address offset: 0x008

Reset value: 0x0000 0000

This register can be written by a software agent with secure privileged attribute in order to individually lock, for example at boot time, the secure privileged attributes of any DMA channel/resource (to lock the setting of LPDMA_SECCFGR, LPDMA_PRIVCFGR for any channel x, for example at boot time).

A read access may be privileged or unprivileged, secure or nonsecure.

Note: If TZEN = 0, this register cannot be written.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
								rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LOCKx**: lock the configuration of LPDMA_SECCFGR.SECx, LPDMA_PRIVCFGR.PRIVx until a global DMA reset (x = 7 to 0)

This bit is cleared after reset and, once set, it cannot be reset until a global DMA reset.

0: secure privilege configuration of the channel x is writable.

1: secure privilege configuration of the channel x is not writable.

16.8.4 LPDMA nonsecure masked interrupt status register (LPDMA_MISR)

Address offset: 0x00C

Reset value: 0x0000 0000

This register is a read register.

This is a nonsecure register, containing the masked interrupt status bit MISx for each nonsecure channel x (channel x configured with LPDMA_SECCFGR.SECx = 0). It is a logical OR of all the flags of LPDMA_CxSR, each source flag being enabled by the corresponding interrupt enable bit of LPDMA_CxCR.

Every bit is de-asserted by hardware when writing 1 to the corresponding flag clear bit in LPDMA_CxFCR.

If a channel x is in secure state (LPDMA_SECCFGR.SECx = 1), a read access to the masked interrupt status bit MISx of this channel x returns zero.

This register may mix privileged and unprivileged information, depending on the privileged state of each channel LPDMA_PRIVCFGR.PRIVx. A privileged software can read the full nonsecure interrupt status. An unprivileged software is restricted to read the status of unprivileged (and nonsecure) channel(s), other privileged bit fields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **MISx**: masked interrupt status of channel x (x = 7 to 0)

0: no interrupt occurred on nonsecure channel x

1: an interrupt occurred on nonsecure channel x

16.8.5 LPDMA secure masked interrupt status register (LPDMA_SMISR)

Address offset: 0x010

Reset value: 0x0000 0000

This is a secure read register, containing the masked interrupt status bit MISx for each secure channel x (LPDMA_SECCFGR.SECx = 1). It is a logical OR of all the LPDMA_CxSR flags, each source flag being enabled by the corresponding LPDMA_CxCR interrupt enable bit.

Every bit is de-asserted by hardware when securely writing 1 to the corresponding LPDMA_CxFCR flag clear bit.

This register does not contain any information about a nonsecure channel.

This register can mix privileged and unprivileged information, depending on the privileged state of each channel LPDMA_PRIVCFGR.PRIVx. A privileged software can read the full secure interrupt status. An unprivileged software is restricted to read the status of unprivileged and secure channels, other privileged bit fields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **MISx**: masked interrupt status of the secure channel x (x = 7 to 0)

0: no interrupt occurred on the secure channel x

1: an interrupt occurred on the secure channel x

16.8.6 LPDMA channel x linked-list base address register (LPDMA_CxLBAR)

Address offset: 0x050 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register must be written by a privileged software. It is either privileged readable or not, depending on the privileged state of the channel x LPDMA_PRIVCFGR.PRIVx.

This register is either secure or nonsecure depending on the secure state of the channel x (LPDMA_SECCFGR.SECx).

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1.

This channel-based register is the linked-list base address of the memory region, for a given channel x, from which the LLIs describing the programmed sequence of the LPDMA transfers, are conditionally and automatically updated.

This 64-Kbyte aligned channel x linked-list base address is offset by the 16-bit LPDMA_CxLLR register that defines the word-aligned address offset for each LLI.

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

Bits 31:16 **LBA[31:16]**: linked-list base address of LPDMA channel x

Bits 15:0 Reserved, must be kept at reset value.

16.8.7 LPDMA channel x flag clear register (LPDMA_CxFCR)

Address offset: 0x05C+ 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This is a write register, secure or nonsecure depending on the secure state of channel x (LPDMA_SECCFGR.SECx) and privileged or unprivileged, depending on the privileged state of the channel x (LPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	w	w	w	w	w	w	w								

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **TOF**: trigger overrun flag clear
0: no effect
1: clears the corresponding TOF flag

Bit 13 **SUSPF**: completed suspension flag clear
0: no effect
1: corresponding SUSPF flag cleared

Bit 12 **USEF**: user setting error flag clear
0: no effect
1: corresponding USEF flag cleared

Bit 11 **ULEF**: update link transfer error flag clear
0: no effect
1: corresponding ULEF flag cleared

Bit 10 **DTEF**: data transfer error flag clear
0: no effect
1: corresponding DTEF flag cleared

Bit 9 **HTF**: half-transfer flag clear
0: no effect
1: corresponding HTF flag cleared

Bit 8 **TCF**: transfer complete flag clear
0: no effect
1: corresponding TCF flag cleared

Bits 7:0 Reserved, must be kept at reset value.

16.8.8 LPDMA channel x status register (LPDMA_CxSR)

Address offset: 0x060 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0001

This is a read register, reporting the channel status.

This register is secure or nonsecure, depending on the secure state of channel x (LPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of the channel (LPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLEF
	r	r	r	r	r	r	r								r

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **TOF**: trigger overrun flag clear
0: no effect
1: clears the corresponding TOF flag

Bit 13 **SUSPF**: completed suspension flag
0: no completed suspension event
1: a completed suspension event occurred

Bit 12 **USEF**: user setting error flag
0: no user setting error event
1: a user setting error event occurred

Bit 11 **ULEF**: update link transfer error flag
0: no update link transfer error event
1: a master bus error event occurred while updating a linked-list register from memory

Bit 10 **DTEF**: data transfer error flag
0: no data transfer error event
1: a master bus error event occurred on a data transfer

Bit 9 **HTF**: half-transfer flag
0: no half-transfer event
1: a half-transfer event occurred
A half-transfer event is a half-block transfer that occurs when half of the bytes of the source block size (rounded-up integer of LPDMA_CxBR1.BNDT[15:0] / 2) has been transferred to the destination.

Bit 8 **TCF**: transfer complete flag
0: no transfer complete event
1: a transfer complete event occurred
A transfer complete event is a block transfer complete or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode LPDMA_CxTR2.TCEM[1:0].

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **IDLEF**: idle flag
0: channel not in idle state
1: channel in idle state
This idle flag is de-asserted by hardware when the channel is enabled (LPDMA_CxCR.EN = 1) with a valid channel configuration (no USEF to be immediately reported).
This idle flag is asserted after hard reset or by hardware when the channel is back in idle state (in suspended or disabled state).

16.8.9 LPDMA channel x control register (LPDMA_CxCR)

Address offset: 0x64 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (LPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of the channel x (LPDMA_PRIVCFGR.PRIVx).

This register is used to control a channel (activate, suspend, abort or disable it).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	Res.	LSM
								rw	rw						rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	Res.	SUSP	RESET	EN
	rw	rw	rw	rw	rw	rw	rw						rw	w	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **PRIO[1:0]:** priority level of the channel x LPDMA transfer versus others
00: low priority, low weight
01: low priority, mid weight
10: low priority, high weight
11: high priority
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 21:17 Reserved, must be kept at reset value.

Bit 16 **LSM**: Link step mode
0: channel executed for the full linked-list and completed at the end of the last LLI (LPDMA_CxLLR = 0). The 16 low-significant bits of the link address are null (LA[15:0] = 0) and all the update bits are null (UT1 = UB1 = UT2 = USA = UDA = ULL = 0). Then LPDMA_CxBR1.BNDT[15:0] = 0.
1: channel executed **once** for the current LLI
First the block transfer is executed as defined by the current internal register file until LPDMA_CxBR1.BNDT[15:0] = 0). Secondly the next linked-list data structure is conditionally uploaded from memory as defined by LPDMA_CxLLR. Then channel execution is completed.
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOIE**: trigger overrun interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 13 **SUSPIE**: completed suspension interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 12 **USEIE**: user setting error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 11 **ULEIE**: update link transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 10 **DTEIE**: data transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 9 **HTIE**: half-transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 8 **TCIE**: transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **SUSP:** suspend
Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 2. Else:
Software must write 1 in order to suspend an active channel (with an ongoing DMA transfer over its master ports).
The software must write 0 in order to resume a suspended channel, following the programming sequence detailed in *Figure 50*.
0: write: resume channel, read: channel not suspended
1: write: suspend channel, read: channel suspended.

Bit 1 **RESET:** reset
This bit is write only. Writing 0 has no impact. Writing 1 implies the reset of the following: the channel internal state, SUSP and EN bits (whatever is written receptively in bit 2 and bit 0).
The reset is effective when the channel is in steady state, meaning one of the following:
- active channel in suspended state (LPDMA_CxSR.SUSPF = 1 and LPDMA_CxSR.IDLEF = LPDMA_CxCR.EN = 1)
- channel in disabled state (LPDMA_CxSR.IDLEF = 1 and LPDMA_CxCR.EN = 0).
After writing a RESET, to continue using this channel, the user must explicitly reconfigure the channel including the hardware-modified configuration registers (LPDMA_CxBR1, LPDMA_CxSAR and LPDMA_CxDAR) before enabling again the channel (see the programming sequence in *Figure 51*).
0: no channel reset
1: channel reset

Bit 0 **EN:** enable
Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 0. Else:
this bit is de-asserted by hardware when there is a transfer error (master bus error or user setting error) or when there is a channel transfer complete (channel ready to be configured, for example: if LSM = 1 at the end of a single execution of the LLI).
Else, this bit can be asserted by software.
Writing 0 into this EN bit is ignored.
0: write: ignored, read: channel disabled
1: write: enable channel, read: channel enabled

16.8.10 LPDMA channel x transfer register 1 (LPDMA_CxTR1)

Address offset: 0x090 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (LPDMA_SECCFGR.SECx) except for secure DSEC and SSEC, privileged or unprivileged, depending on the privileged state of the channel x in LPDMA_PRIVCFGR.PRIVx.

This register controls the transfer of a channel x.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1.

This register must be written when the channel is completed. Then the hardware has deasserted LPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by LPDMA from the memory if LPDMA_CxLLR.UT1 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DSEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DINC	Res.	DDW_LOG2[1:0]	
rw												rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSEC	Res.	Res.	Res.	PAM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SINC	Res.	SDW_LOG2[1:0]	
rw				rw								rw		rw	rw

Bit 31 **DSEC**: security attribute of the LPDMA transfer to the destination

If LPDMA_SECCFGR.SECx = 1 and the access is secure:

0: LPDMA transfer nonsecure

1: LPDMA transfer secure

This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when LPDMA_SECCFGR.SECx = 1. A secure write is ignored when LPDMA_SECCFGR.SECx = 0.

When LPDMA_SECCFGR.SECx is de-asserted, this DSEC bit is also de-asserted by hardware (on a secure reconfiguration of the channel as nonsecure), and the LPDMA transfer to the destination is nonsecure.

Bits 30:20 Reserved, must be kept at reset value.

Bit 19 **DINC**: destination incrementing single

0: fixed single

1: contiguously incremented single

The destination address, pointed by LPDMA_CxDAR, is kept constant after a single transfer, or is incremented by the offset value corresponding to a contiguous data after a single transfer.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **DDW_LOG2[1:0]**: binary logarithm of the destination data width of a single in bytes
00: byte
01: half-word (2 bytes)
10: word (4 bytes)
11: user setting error reported and no transfer issued

Note: Setting a 8-byte data width causes a user setting error to be reported and none transfer is issued.
A destination single transfer must have an aligned address with its data width (start address LPDMA_CxDAR[2:0] versus DDW_LOG2[1:0]). Otherwise a user setting error is reported and none transfer is issued.

Bit 15 **SSEC**: security attribute of the LPDMA transfer from the source
If LPDMA_SECCFGR.SECx = 1 and the access is secure:
0: LPDMA transfer nonsecure
1: LPDMA transfer secure
This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when LPDMA_SECCFGR.SECx =1 . A secure write is ignored when LPDMA_SECCFGR.SECx = 0.
When LPDMA_SECCFGR.SECx is de-asserted, this SSEC bit is also de-asserted by hardware (on a secure reconfiguration of the channel as nonsecure), and the LPDMA transfer from the source is nonsecure.

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **PAM**: padding/alignment mode
If DDW_LOG2[1:0] = SDW_LOG2[1:0]: if the data width of a single destination transfer is equal to the data width of a single source transfer, this bit is ignored.
Else, in the following enumerated values, the condition PAM_1 is when destination data width is higher that source data width, and the condition PAM_2 is when destination data width is higher than source data width.
Condition: PAM_1
0: source data is transferred as right aligned, padded with 0s up to the destination data width
1: source data is transferred as right aligned, sign extended up to the destination data width
Condition: PAM_2
0: source data is transferred as right aligned, left-truncated down to the destination data width
1: source data is transferred as left-aligned, right-truncated down to the destination data width

Bits 10:4 Reserved, must be kept at reset value.

Bit 3 **SINC**: source incrementing single
0: fixed single
1: contiguously incremented single
The source address, pointed by LPDMA_CxSAR, is kept constant after a single transfer or is incremented by the offset value corresponding to a contiguous data after a single transfer.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SDW_LOG2[1:0]**: binary logarithm of the source data width of a single in bytes
00: byte
01: half-word (2 bytes)
10: word (4 bytes)
11: user setting error reported and no transfer issued

Note: Setting a 8-byte data width causes a user setting error to be reported and none transfer is issued.
a source block size must be a multiple of the source data width (LPDMA_CxBR1.BNDT[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and no transfer is issued.
A source single transfer must have an aligned address with its data width (start address LPDMA_CxSAR[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and none transfer is issued.

16.8.11 LPDMA channel x transfer register 2 (LPDMA_CxTR2)

Address offset: 0x094 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (LPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (LPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (the hardware de-asserted LPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by LPDMA from the memory, if LPDMA_CxLLR.UT2 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	Res.	Res.	TRIGSEL[4:0]				
rw	rw					rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGM[1:0]		Res.	Res.	BREQ	Res.	SWREQ	Res.	Res.	Res.	REQSEL[5:0]					
rw	rw			rw		rw				rw	rw	rw	rw	rw	rw

Bits 31:30 **TCEM[1:0]**: transfer complete event mode

These bits define the transfer granularity for the transfer complete and half-transfer complete events generation.

00: at block level (when LPDMA_CxBR1.BNDT[15:0] = 0): the complete (and the half) transfer event is generated at the (respectively half of the) end of a block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with LPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half-transfer event is generated.

01: same as 00

10: at LLI level: the complete transfer event is generated at the end of the LLI transfer, including the update of the LLI if any. The half-transfer event is generated at the half of the LLI data transfer (the LLI data transfer being a block transfer), if any data transfer.

Note: If the initial LLI_0 data transfer is null/void (i.e. directly programmed by the internal register file with LPDMA_CxBR1.BNDT[15:0] = 0), then the half-transfer event is not generated, and the transfer complete event is generated when is completed the loading of the LLI_1.

11: at channel level: the complete transfer event is generated at the end of the last LLI transfer. The half-transfer event is generated at the half of the data transfer of the last LLI. The last LLI updates the link address LPDMA_CxLLR.LA[15:2] to zero and clears all the LPDMA_CxLLR update bits (UT1, UT2, UB1, USA, UDA and ULL). If the channel transfer is continuous/infinite, no event is generated.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **TRIGPOL[1:0]**: trigger event polarity

These bits define the polarity of the selected trigger event input defined by TRIGSEL[4:0].

00: no trigger (masked trigger event)

01: trigger on the rising edge

10: trigger on the falling edge

11: same as 00

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **TRIGSEL[4:0]**: trigger event input selection

These bits select the trigger event input of the LPDMA transfer (as per *Section 16.3.5*), with an active trigger event if TRIGPOL[1:0] = 00.

Bits 15:14 **TRIGM[1:0]**: trigger mode

These bits define the transfer granularity for its conditioning by the trigger.

If the channel x is enabled (LPDMA_CxCR.EN asserted) with TRIGPOL[1:0] = 0b00 or 0b11, these TRIGM[1:0] bits are ignored.

Else, a DMA transfer is conditioned by at least one trigger hit:

00: at block level: the first single read of each block transfer is conditioned by one hit trigger.

01: same as 00

10: at link level: a LLI link transfer is conditioned by one hit trigger. The LLI data transfer (if any) is not conditioned.

11: at programmed single level: each programmed single read is conditioned by one hit trigger.

The LPDMA monitoring of a trigger for channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (TRIGPOL[1:0] = 0b01 or respectively TRIGPOL[1:0] = 0b10).

The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer; and if a new trigger is detected then, this hit is internally memorized to grant the next transfer, as long as the defined rising or falling edge is not modified, and the TRIGSEL[4:0] is not modified, and the channel is enabled.

Transferring a next LLI_{n+1} that updates the LPDMA_CxTR2 with a new value for any of TRIGSEL[4:0] or TRIGPOL[1:0], resets the monitoring, trashing the memorized hit of the formerly defined LLI_n trigger.

After a first new trigger hit_{n+1} is memorized, if another second trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized, and a trigger overrun flag is reported (LPDMA_CxSR.TOF = 1), an interrupt is generated if enabled (LPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **BREQ**: block hardware request

If the channel x is activated (LPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a single level.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a block level (see *Section 16.3.4*).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **SWREQ**: software request

This bit is internally taken into account when LPDMA_CxCR.EN is asserted.

0: no software request. The selected hardware request REQSEL[5:0] is taken into account.

1: software request for a memory-to-memory transfer. The default selected hardware request as per REQSEL[5:0] is ignored.

Bits 8:6 Reserved, must be kept at reset value.

Bits 5:0 **REQSEL[5:0]**: DMA hardware request selection

These bits are ignored if channel x is activated (LPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else, the selected hardware request is internally taken into account as per *Section 16.3.3*.

Caution: The user must not assign a same input hardware request (same REQSEL[5:0] value) to different active DMA channels (LPDMA_CxCR.EN = 1 and LPDMA_CxTR2.SWREQ = 0 for these channels). DMA is not intended to hardware support the case of simultaneous enabled channels incorrectly configured with a same hardware peripheral request signal, and there is no user setting error reporting.

16.8.12 LPDMA channel x block register 1 (LPDMA_CxBR1)

Address offset: 0x098 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (LPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (LPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has de-asserted LPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if LPDMA_CxLLR.UB1 = 1, this register is automatically updated by DMA from the next LLI in memory.
- If LPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed values for the field BNDT[15:0].
- If all the update bits LPDMA_CxLLR.Uxx are null and if LPDMA_CxLLR.LA[15:0] # 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for BNDT[15:0] after each execution of this final LLI
- If LPDMA_CxLLR = 0, this register and BNDT[15:0] are kept as null, channel x is completed.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if LPDMA_CxLLR.UB1 = 1, this field is updated with the LLI in the memory.
- if LPDMA_CxLLR.UB1 = 0 and if there is at least one non null Uxx update bit, this field is internally restored to the programmed value.
- if all LPDMA_CxLLR.Uxx = 0 and if LPDMA_CxLLR.LA[15:0] = 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
- if LPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

Note: A non-null source block size must be a multiple of the source data width (BNDT[2:0] versus LPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

16.8.13 LPDMA channel x source address register (LPDMA_CxSAR)

Address offset: 0x09C + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (LPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (LPDMA_PRIVCFGR.PRIVx).

This register configures the source start address of a transfer.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next single transfer from the source.

This register must be written when the channel is completed (then the hardware has deasserted LPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by LPDMA from the memory if LPDMA_CxLLR.USA = 1.

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

Bits 31:0 **SA[31:0]**: source address

This field is the pointer to the address from which the next data is read.

During the channel activity, depending on the source addressing mode (LPDMA_CxTR1.SINC), this field is either kept fixed or incremented by the data width (LPDMA_CxTR1.SDW_LOG2[1:0]) after each single source data, reflecting the next address from which data is read.

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by LPDMA from the memory, provided the LLI is set with LPDMA_CxLLR.USA = 1.

Note: A source address must be aligned with the programmed data width of a source single (SA[32:0] versus LPDMA_CxTR1.SDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

16.8.14 LPDMA channel x destination address register (LPDMA_CxDAR)

Address offset: 0x0A0 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (LPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (LPDMA_PRIVCFGR.PRIVx).

This register configures the destination start address of a transfer.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next single transfer to the destination.

This register must be written when the channel is completed (then the hardware has deasserted LPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by LPDMA from the memory if LPDMA_CxLLR.UDA = 1.

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

Bits 31:0 **DA[31:0]**: destination address

This field is the pointer to the address from which the next data is written.

During the channel activity, depending on the destination addressing mode (LPDMA_CxTR1.DINC), this field is kept fixed or incremented by the data width (LPDMA_CxTR1.DDW_LOG2[1:0]) after each single destination data, reflecting the next address from which data is written.

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by DMA from the memory, provided the LLI is set with LPDMA_CxLLR.UDA = 1.

Note: A destination address must be aligned with the programmed data width of a destination single (DA[2:0] versus LPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

16.8.15 LPDMA channel x linked-list address register (LPDMA_CxLLR)

Address offset: 0x0CC + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (LPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (LPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted LPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by LPDMA from the memory if LPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw											rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update LPDMA_CxTR1 from memory

This bit controls the update of the LPDMA_CxTR1 register from the memory during the link transfer.

0: no LPDMA_CxTR1 update

1: LPDMA_CxTR1 update

Bit 30 **UT2**: Update LPDMA_CxTR2 from memory

This bit controls the update of the LPDMA_CxTR2 register from the memory during the link transfer.

0: no LPDMA_CxTR2 update

1: LPDMA_CxTR2 update

Bit 29 **UB1**: Update LPDMA_CxBR1 from memory

This bit controls the update of the LPDMA_CxBR1 register from the memory during the link transfer.

0: no LPDMA_CxBR1 update from memory and internally restored to the previous programmed value

1: LPDMA_CxBR1 update

Bit 28 **USA**: update LPDMA_CxSAR from memory

This bit controls the update of the LPDMA_CxSAR register from the memory during the link transfer.

0: no LPDMA_CxSAR update

1: LPDMA_CxSAR update

Bit 27 **UDA**: Update LPDMA_CxDAR register from memory

This bit is used to control the update of the LPDMA_CxDAR register from the memory during the link transfer.

0: no LPDMA_CxDAR update

1: LPDMA_CxDAR update

Bits 26:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update LPDMA_CxLLR register from memory

This bit is used to control the update of the LPDMA_CxLLR register from the memory during the link transfer.

0: no LPDMA_CxLLR update

1: LPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure

If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list DMA register file.

Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list DMA internal register file (LPDMA_CxTR1, LPDMA_CxTR2, LPDMA_CxBR1, LPDMA_CxSAR, LPDMA_CxDAR and LPDMA_CxLLR).

Caution: The user must program this pointer not to exceed the 64-Kbyte addressable space defined by the link base address register LPDMA_CxLBAR. The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

Bits 1:0 Reserved, must be kept at reset value.

16.8.16 LPDMA register map

Table 139. LPDMA register map and reset values

Offset	Register name	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
0x000	LPDMA_SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value																									0	0	0	0	0	0	0	0
0x004	LPDMA_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value																									0	0	0	0	0	0	0	0
0x008	LPDMA_ RCFGLOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
	Reset value																									0	0	0	0	0	0	0	0
0x00C	LPDMA_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																									0	0	0	0	0	0	0	0
0x010	LPDMA_SMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																									0	0	0	0	0	0	0	0
0x014 - 0x04C	Reserved	Res.																															
0x050+ 0x080 * x (x = 0 to 7)	LPDMA_CxLBAR	LBA[31:16]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x05C+ 0x080 * x (x = 0 to 7)	LPDMA_CxFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0	0	0	0	0	0	0								
0x060+ 0x080 * x (x = 0 to 7)	LPDMA_CxSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLEF
	Reset value																		0	0	0	0	0	0	0								1

Table 139. LPDMA register map and reset values (continued)

Offset	Register name	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
0x064+ 0x080 * x (x = 0 to 7)	LPDMA_CxCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	Res.	LSM	Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	Res.	SUSP	RESET	EN
	Reset value									0	0						0		0	0	0	0	0	0	0						0	0	0
0x090+ 0x080 * x (x = 0 to 7)	LPDMA_CxTR1	DSEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DINC	Res.	DDW_LOG2[1:0]		SSEC	Res.	Res.	Res.	PAM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SINC	Res.	SDW_LOG2[1:0]	
	Reset value	0												0		0	0	0				0								0		0	0
0x094+ 0x080 * x (x = 0 to 7)	LPDMA_CxTR2	TCEM [1:0]		Res.	Res.	Res.	Res.	TRIGPOL [1:0]		Res.	Res.	TRIGSEL [4:0]					Res.	TRIGM [1:0]		Res.	Res.	Res.	BREQ	SWREQ	Res.	Res.	Res.	REQSEL [5:0]					
	Reset value	0	0					0	0			0	0	0	0	0		0	0			0	0	0					0	0	0	0	0
0x098+ 0x080 * x (x = 0 to 7)	LPDMA_CxBR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BNDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x09C+ 0x080 * x (x = 0 to 7)	LPDMA_CxSAR	SA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0+ 0x080 * x (x = 0 to 7)	LPDMA_CxDAR	DA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0CC+ 0x080 * x (x = 0 to 7)	LPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

17 Nested vectored interrupt controller (NVIC)

The NVIC is a Arm® Cortex® embedded interrupt controller that supports low-latency interrupt processing.

17.1 NVIC main features

- 16 programmable priority levels (4 bits of interrupt priority used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of system control registers

The NVIC and the processor core interface are closely coupled, enabling low-latency interrupt processing and efficient processing of late arriving interrupts.

The NVIC registers are banked across secure and nonsecure states.

All interrupts including the core exceptions are managed by the NVIC.

17.2 Interrupt and exception vectors

The grey rows in *Table 140* indicate vectors without specific position.

Table 140. Vector table

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000 0000
-	-4	Fixed	Reset	Reset	0x0000 0004
-14	-2	Fixed	NMI	Nonmaskable interrupt. - RCC clock security system (HSECSS) - FLASH ECC double error detection - RAMCFG SRAM2 parity error	0x0000 0008
-13	-1	Fixed	Secure HardFault	Hard fault when AIRCR.BFHFNMINS = 1	0x0000 000C
	-1	Fixed		Hard fault when AIRCR.BFHFNMINS = 0	
	-1	Fixed	Nonsecure HardFault	Hard fault. All classes of fault	
-12	0	Settable	MemManage	Memory management	0x0000 0010
-11	1	Settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
-10	2	Settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
-9	3	Settable	SecureFault	Secure fault	0x0000 001C
-	-	-	-	Reserved	0x0000 0020 - 0x0000 0028

Table 140. Vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
-5	4	Settable	SVCall	System service call via SWI instruction	0x0000 002C
-4	5	Settable	Debug monitor	Debug monitor	0x0000 0030
-	-	-	-	Reserved	0x0000 0034
-2	6	Settable	PendSV	Pendable request for system service	0x0000 0038
-1	7	Settable	SysTick	System tick timer	0x0000 003C
0	8	Settable	-	Reserved	0x0000 0040
1	9	Settable	EXTI 16 (PVD)	EXTI line 16 interrupt, Power voltage monitor	0x0000 0044
2	10	Settable	RTC	RTC nonsecure global interrupts	0x0000 0048
3	11	Settable	RTC_S	RTC secure global interrupts	0x0000 004C
4	12	Settable	TAMP	Tamper global interrupt	0x0000 0050
5	13	Settable	RAMCFG	RAM configuration global interrupt	0x0000 0054
6	14	Settable	FLASH	Flash nonsecure interface global interrupt Flash ECC single error correction interrupt	0x0000 0058
7	15	Settable	FLASH_S	Flash secure interface global interrupt	0x0000 005C
8	16	Settable	GTZC1_TZIC	GTZC1_TZIC global interrupt	0x0000 0060
9	17	Settable	RCC	RCC nonsecure global interrupt	0x0000 0064
10	18	Settable	RCC_S	RCC secure global interrupt	0x0000 0068
11	19	Settable	EXTI0 (GPIO)	EXTI line- 0 interrupt, GPIO 0	0x0000 006C
12	20	Settable	EXTI1 (GPIO)	EXTI line 1 interrupt, GPIO 1	0x0000 0070
13	21	Settable	EXTI2 (GPIO)	EXTI line 2 interrupt, GPIO 2	0x0000 0074
14	22	Settable	EXTI3 (GPIO)	EXTI line3 interrupt , GPIO 3	0x0000 0078
15	23	Settable	EXTI4 (GPIO)	EXTI line 4 interrupt, GPIO 4	0x0000 007C
16	24	Settable	EXTI5 (GPIO)	EXTI line 5 interrupt, GPIO 5	0x0000 0080
17	25	Settable	EXTI6 (GPIO)	EXTI line 6 interrupt, GPIO 6	0x0000 0084
18	26	Settable	EXTI7 (GPIO)	EXTI line 7 interrupt, GPIO 7	0x0000 0088
19	27	Settable	EXTI8 (GPIO)	EXTI line 8 interrupt, GPIO 8	0x0000 008C
20	28	Settable	EXTI9 (GPIO)	EXTI line 9 interrupt, GPIO 9	0x0000 0090
21	29	Settable	EXTI10 (GPIO)	EXTI line 10 interrupt, GPIO 10	0x0000 0094
22	30	Settable	EXTI11 (GPIO)	EXTI line 11 interrupt, GPIO 11	0x0000 0098
23	31	Settable	EXTI12 (GPIO)	EXTI line 12 interrupt, GPIO 12	0x0000 009C
24	32	Settable	EXTI13 (GPIO)	EXTI line 13 interrupt, GPIO 13	0x0000 00A0
25	33	Settable	EXTI14 (GPIO)	EXTI line 14 interrupt, GPIO 14	0x0000 00A4
26	34	Settable	EXTI15 (GPIO)	EXTI line 15 interrupt, GPIO 15	0x0000 00A8

Table 140. Vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
27	35	Settable	IWDG	Independent watchdog interrupt	0x0000 00AC
28	36	Settable	LPDMA1_CH0	LPDMA1 channel 0 global interrupt	0x0000 00B0
29	37	Settable	LPDMA1_CH1	LPDMA1 channel 1 global interrupt	0x0000 00B4
30	38	Settable	LPDMA1_CH2	LPDMA1 channel 2 global interrupt	0x0000 00B8
31	39	Settable	LPDMA1_CH3	LPDMA1 channel 3 global interrupt	0x0000 00BC
32	40	Settable	LPDMA1_CH4	LPDMA1 channel 4 global interrupt	0x0000 00C0
33	41	Settable	LPDMA1_CH5	LPDMA1 channel 5 global interrupt	0x0000 00C4
34	42	Settable	LPDMA1_CH6	LPDMA1 channel 6 global interrupt	0x0000 00C8
35	43	Settable	LPDMA1_CH7	LPDMA1 channel 7 global interrupt	0x0000 00CC
36	44	Settable	OTFDEC1(1)	OTFDEC1 global interrupt	0x0000 00D0
37	45	Settable	-	Reserved	0x0000 00D4
38	46	Settable	-	Reserved	0x0000 00D8
39	47	Settable	-	Reserved	0x0000 00DC
40	48	Settable	TIM2	TIM2 global interrupt	0x0000 00E0
41	49	Settable	I2C1_EV	I2C1 event interrupt	0x0000 00E4
42	50	Settable	I2C1_ER	I2C1 error interrupt	0x0000 00E8
43	51	Settable	USART1	USART1 global interrupt	0x0000 00EC
44	52	Settable	LPUART1	LPUART1 global interrupt	0x0000 00F0
45	53	Settable	LPTIM1	LPTIM1 global interrupt	0x0000 00F4
46	54	Settable	LPTIM2	LPTIM2 global interrupt	0x0000 00F8
47	55	Settable	TIM16	TIM16 global interrupt	0x0000 00FC
48	56	Settable	TIM17	TIM17 global interrupt	0x0000 0100
49	57	Settable	I2C3_EV	I2C3 event interrupt	0x0000 0104
50	58	Settable	I2C3_ER	I2C3 error interrupt	0x0000 0108
51	59	Settable	SAI1	SAI global interrupt	0x0000 010C
52	60	Settable	AES	AES global interrupt	0x0000 0110
53	61	Settable	RNG	RNG global interrupt	0x0000 0114
54	62	Settable	FPU	Floating point interrupt	0x0000 0118
55	63	Settable	HASH	HASH global interrupt	0x0000 011C
56	64	Settable	PKA	PKA global interrupt	0x0000 0120
57	65	Settable	SPI3	SPI3 global interrupt	0x0000 0124
58	66	Settable	ICACHE	ICACHE global interrupt	0x0000 0128
59	67	Settable	ADC4	ADC4 global interrupt	0x0000 012C

Table 140. Vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
60	68	Settable	RADIO	2.4 GHz RADIO global interrupt	0x0000 0130
61	69	Settable	WKUP	PWR nonsecure global WKUP pin interrupt	0x0000 0134
62	70	Settable	WKUP_S	PWR secure global WKUP pin interrupt	0x0000 0138
63	71	Settable	RCC_AUDIOSYNC	RCC audio synchronization interrupt	0x0000 013C
64	72	Settable	USB(1)	USB global interrupt	0x0000 0140
65	73	Settable	XSPI1(1)	XSPI1 global interrupt	0x0000 0144

1. Only available on STM32WBA25xx devices

18 Extended interrupts and event controller (EXTI)

The extended interrupts and event controller (EXTI) manages the individual CPU and system wake-up through configurable event inputs. It provides wake-up requests to the power control and generates an interrupt request to the CPU NVIC and events to the CPU event input. For the CPU, an additional event generation block (EVG) is needed to generate the CPU event signal.

The EXTI wake-up requests allow the system to be woken up from Stop modes.

The interrupt request and event request generation can be used also in Run modes.

The EXTI also includes the EXTI mux IO port selection.

18.1 EXTI main features

- 17 input events supported
- All event inputs allow the possibility to wake up the system.
- Events that do not have an associated wake-up flag in the peripheral, have a flag in the EXTI and generate an interrupt to the CPU from the EXTI.
- Events can be used to generate a CPU wake-up event.

The configurable events have the following features:

- Selectable active trigger edge
- Interrupt pending status register bits independent for the rising and falling edge
- Individual interrupt and event generation mask, used for conditioning the CPU wake-up, interrupt and event generation
- Software trigger possibility
- Secure events: The access to control and configuration bits of secure input events can be made secure and/or privileged.
- EXTI IO port selection

18.2 EXTI block diagram

The EXTI consists of a register block accessed via an AHB interface, the event input trigger block, the masking block and EXTI mux, as shown in *Figure 66*:

- the register block contains all the EXTI registers
- the event input trigger block provides event input edge trigger logic
- the masking block provides the event input distribution to the different wake-up, interrupt and event outputs, and their masking
- the EXTI mux provides the IO port selection on to the EXTI event signal.

Figure 66. EXTI block diagram

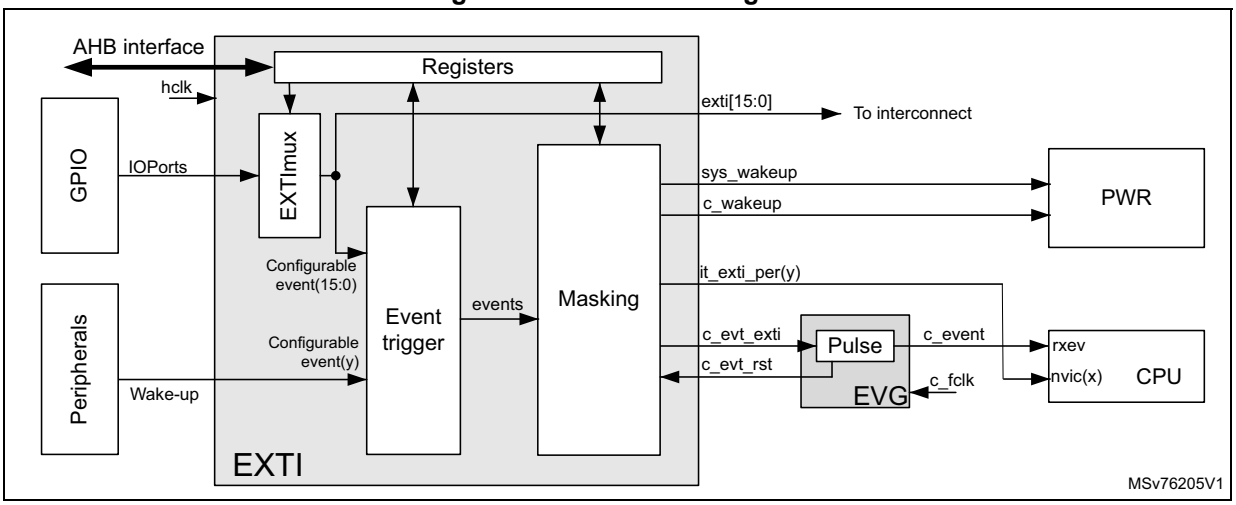


Table 141. EXTI pin overview

Pin name	I/O	Description
AHB interface	I/O	EXTI register bus interface. When one event is configured to enable security, the AHB interface supports secure accesses.
hclk	I	AHB bus clock and EXTI system clock
Configurable event(y)	I	Asynchronous wake-up events from peripherals that do not have an associated interrupt and flag in the peripheral
IOPort(n)	I	GPIOs block IO ports[15:0]
exti[15:0]	O	EXTI GPIO output port to trigger other peripherals
it_exti_per (y)	O	Interrupts to the CPU associated with configurable event (y)
c_evt_exti	O	High-level sensitive event output for CPU, synchronous to hclk
c_evt_rst	I	Asynchronous reset input to clear c_evt_exti
sys_wakeup	O	Asynchronous system wake-up request to PWR for ck_sys and hclk
c_wakeup	O	Wake-up request to PWR for CPU, synchronous to hclk

Table 142. EVG pin overview

Pin name	I/O	Description
c_fclk	I	CPU free running clock
c_evt_in	I	High-level sensitive events input from EXTI, asynchronous to CPU clock
c_event	O	Event pulse, synchronous to CPU clock
c_evt_rst	O	Event reset signal, synchronous to CPU clock

18.2.1 EXTI connections between peripherals and CPU

Some peripherals able to generate wake-up or interrupt events when the system is in Stop mode, are connected to the EXTI.

- Peripheral wake-up signals that generate a pulse or do not have an interrupt status bits in the peripheral, are connected to an EXTI configurable event input. For these events, the EXTI provides a status pending bit that requires to be cleared. It is the EXTI interrupt, associated with the status bit, that interrupts the CPU.
- All GPIO ports input to the EXTI multiplexer make possible the selection of a port pin to wake up the system via a configurable event.

The EXTI configurable event interrupts are connected to the NVIC.

The dedicated EXTI/EVG CPU event is connected to the CPU rxev input.

The EXTI CPU wake-up signals are connected to the PWR and are used to wake up the system and the CPU sub-system bus clocks.

18.2.2 EXTI interrupt/event mapping

The EXTI lines are connected as shown in the table below.

Table 143. EXTI line connections

EXTI line	Line source	Line type
0-15	GPIO	Configurable
16	PVD output	Configurable
17-18	reserved	-

18.3 EXTI functional description

The events features are controlled from register bits as follows:

- Active trigger edge enable
  - by rising edge selection in the *EXTI rising trigger selection register (EXTI_RTSR1)*
  - by falling edge selection in the *EXTI falling trigger selection register (EXTI_FTSR1)*
- Software trigger in the *EXTI software interrupt event register (EXTI_SWIER1)*
- Interrupt pending flag in the

  EXTI rising edge pending register (EXTI_RPR1)

  EXTI falling edge pending register (EXTI_FPR1)
- CPU wake-up and interrupt enable in the

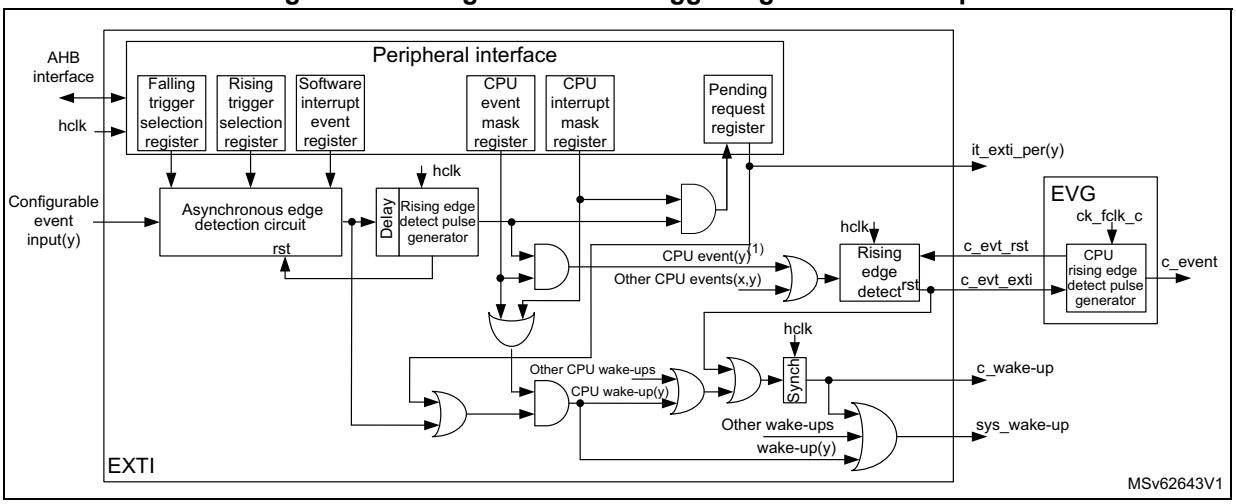
  EXTI lock register (EXTI_LOCKR)
- CPU wake-up and event enable in the

  EXTI CPU wake-up with event mask register (EXTI_EMR1)

18.3.1 EXTI configurable event input wake-up

The figure below is a detailed representation of the logic associated with configurable event inputs that wake up the CPU sub-system bus clocks and generate an EXTI pending flag and interrupt to the CPU, and/or a CPU wake-up event.

Figure 67. Configurable event trigger logic CPU wake-up



1. Only for the input events that support CPU rxev generation c_event.

The software interrupt event register allows configurable events to be triggered by software, writing the corresponding register bit, whatever the edge selection setting.

The configurable event active trigger edge (or both edges) is selected and enabled in the rising/falling edge selection registers.

The CPU has its dedicated wake-up (interrupt) mask register and a dedicated event mask registers. When the event is enabled, it is generated to the CPU. All events for the CPU are ordered together into a single CPU event signal. The event pending registers (EXTI_RPR and EXTI_FPR) are not set for an unmasked CPU event.

The configurable events have unique interrupt pending request registers. The pending register is only set for an unmasked interrupt. Each configurable event provides a common interrupt to the CPU. The configurable event interrupts must be acknowledged by software in the EXTI_RPR and/or EXTI_FPR registers.

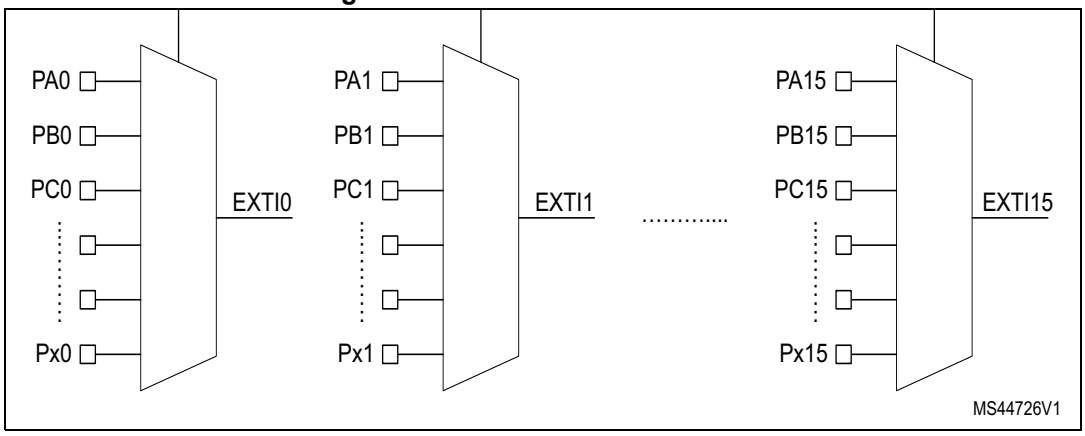
When a CPU wake-up (interrupt) or CPU event is enabled, the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees that the EXTI hclk clock is woken up before the asynchronous edge detection circuit is reset.

Note: A detected configurable event interrupt pending request can be cleared by the CPU with the correct access permission. The system is not able to enter into low-power modes as long as an interrupt pending request is active.

18.3.2 EXTI mux selection

The EXTI mux allows the selection of GPIOs as interrupts and wake-up. GPIOs are connected via 16 EXTI mux lines to the first 16 EXTI events as configurable event. The selection of GPIO port as EXTI mux output is controlled in the *EXTI security configuration register (EXTI_SECCFGR1)*.

Figure 68. EXTI mux GPIO selection



The EXTI mux outputs are available as output signals from the EXTI to trigger other peripherals, whatever the masking in EXTI_IMR and EXTI_EMR registers.

18.4 EXTI functional behavior

The configurable events are enabled by enabling at least one of the trigger edges.

Once an event input is enabled, the CPU wake-up generation is conditioned by the CPU interrupt mask and CPU event mask.

Table 144. Masking functionality

CPU interrupt enable (in EXTI_IMR.IMn)	CPU event enable (in EXTI_EMR.EMn)	Configurable event inputs (in EXTI_RPR.RPIFn and EXTI_FPR.FPIFn)	EXTI(n) interrupt	CPU event	CPU wake-up
0	0	No	Masked	Masked	Masked
	1			Yes	Yes
1	0	Status latched	Yes	Masked	Yes
	1			Yes	Yes

For configurable event inputs, when the enabled edges occur on the event input, an event request is generated. When the associated CPU interrupt is unmasked, the corresponding pending bits EXTI_RPR.RPIFn and/or EXTI_FPR.FPIFn is/are set: the CPU sub-system is woken up and the CPU interrupt signal is activated. The EXTI_RPR.RPIFn and/or EXTI_FPR.FPIFn pending bits must be cleared by software writing it to 1. This action clears the CPU interrupt.

For the configurable event inputs, an event request can be generated by software when writing a 1 in the software interrupt/event register EXTI_SWIER, allowing the generation of a rising edge on the event. The rising edge event pending bit is set in EXTI_RPR, whatever the setting in EXTI_RTSR.

18.5 EXTI event protection

The EXTI is able to protect event register bits from being modified by nonsecure and unprivileged accesses. The protection is individually activated per input event via the register bits in EXTI_SECCFGR and EXTI_PRIVCFGR. At EXTI level, the protection consists in preventing the following unauthorized write access:

- Change the settings of the secure and/or privileged configurable events.
- Change the masking of the secure and/or privileged input events.
- Clear pending status of the secure and/or privileged input events.

Table 145. Register protection overview

Register name	Access type	Protection(1)(2)
EXTI_RTSR	RW	Security and privilege can be bit-wise enabled in EXTI_SECCFGR and EXTI_PRIVCFGR.
EXTI_FTSR	RW	
EXTI_SWIER	RW	
EXTI_RPR	RW	
EXTI_FPR	RW	
EXTI_SECCFGR	RW	Always secure. Privilege can be bit-wise enabled in EXTI_PRIVCFGR.
EXTI_PRIVCFGR	RW	Always privileged. Security can be bit-wise enabled in EXTI_SECCFGR.
EXTI_EXTICRn	RW	Security and privilege can be bit-wise enabled in EXTI_SECCFGR and EXTI_PRIVCFGR.
EXTI_LOCKR	RW	Always secure
EXTI_IM	RW	Security and privilege can be bit-wise enabled in EXTI_SECCFGR and EXTI_PRIVCFGR.
EXTI_EMR	RW	
EXTI_HWCFGR	R	Nonsecure unprivileged
EXTI_VER	R	
EXTI_ID	R	
EXTI_SID	R	

1. Security is enabled with the individual input event (EXTI_SECCFGR register).
2. Privilege is enabled with the individual Input event (EXTI_PRIVCFGR register).

18.5.1 EXTI security protection

When security is enabled for an input event, the associated input event configuration and control bits can only be modified and read by a secure access. A nonsecure write access is discarded and a read returns 0.

When input events are nonsecure, the security is disabled. The associated input event configuration and control bits can be modified and read by a secure access and nonsecure access.

The security configuration in registers EXTI_SECCFGR can be globally locked after reset by EXTI_LOCKR.LOCK.

18.5.2 EXTI privilege protection

When privilege is enabled for an input event, the associated input event configuration and control bits can only be modified and read by a privileged access. An unprivileged write access is discarded and a read returns 0.

When input events are unprivileged, the privilege is disabled. The associated input event configuration and control bits can be modified and read by a privileged access and unprivileged access.

The privileged configuration in registers EXTI_PRIVCFGR can be globally locked after reset by EXTI_LOCKR.LOCK.

18.6 EXTI registers

The EXTI register map is divided in sections, as indicated in *Table 146*.

Table 146. EXTI register map sections

Address offset	Description
0x000 - 0x01C	General configurable event configuration
0x060 - 0x06C	EXTI IO port mux selection
0x070	EXTI protection lock configuration
0x080 - 0x0BC	CPU input event configuration

All the registers can be accessed with word (32-bit), half-word (16-bit) and byte (8-bit) access.

18.6.1 EXTI rising trigger selection register (EXTI_RTSR1)

Address offset: 0x000

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **RT[16:0]:** Rising trigger event configuration bit of configurable event input x[(1)] (x = 0 to 16)
When EXTI_SECCFGR.SECx is disabled, RTx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, RTx can only be accessed with secure access. Nonsecure write to this bit x is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, RTx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, RTx can only be accessed with privileged access. Unprivileged write to this bit x is discarded, unprivileged read returns 0.
0: Rising trigger disabled (for event and interrupt) for input line
1: Rising trigger enabled (for event and interrupt) for input line

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

18.6.2 EXTI falling trigger selection register (EXTI_FTSR1)

Address offset: 0x004

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **FT[16:0]:** Falling trigger event configuration bit of configurable event input x [1] (x = 0 to 16)

When EXTI_SECCFGR.SECx is disabled, FTx can be accessed with nonsecure and secure access.

When EXTI_SECCFGR.SECx is enabled, FTx can only be accessed with secure access. Nonsecure write to this FTx is discarded, nonsecure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, FTx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, FTx can only be accessed with privileged access. Unprivileged write to this FTx is discarded, unprivileged read returns 0.

0: Falling trigger disabled (for event and Interrupt) for input line

1: Falling trigger enabled (for event and Interrupt) for input line.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

18.6.3 EXTI software interrupt event register (EXTI_SWIER1)

Address offset: 0x008

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **SWI[16:0]:** Software interrupt on event x (x = 0 to 16)

A software interrupt is generated independently from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.

When EXTI_SECFGR.SECx is disabled, SWIx can be accessed with nonsecure and secure access.

When EXTI_SECCFGR.SECx is enabled, SWIx can only be accessed with secure access. Nonsecure write to this SWI x is discarded, nonsecure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, SWIx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, SWIx can only be accessed with privileged access. Unprivileged write to this SWIx is discarded, unprivileged read returns 0.

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.

0: Writing 0 has no effect.

1: Writing 1 triggers a rising edge event on event x. This bit is auto cleared by hardware.

18.6.4 EXTI rising edge pending register (EXTI_RPR1)

Address offset: 0x00C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF16
															rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPIF15	RPIF14	RPIF13	RPIF12	RPIF11	RPIF10	RPIF9	RPIF8	RPIF7	RPIF6	RPIF5	RPIF4	RPIF3	RPIF2	RPIF1	RPIF0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **RPIF[16:0]:** configurable event inputs x rising edge pending bit (x = 0 to 16)

When EXTI_SECCFGR.SECx is disabled, RPIFx can be accessed with nonsecure and secure access.

When EXTI_SECCFGR.SECx is enabled, RPIFx can only be accessed with secure access. Nonsecure write to this RPIFx is discarded, nonsecure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, RPIFx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, RPIFx can only be accessed with privileged access. Unprivileged write to this RPIFx is discarded, unprivileged read returns 0.

0: No rising edge trigger request occurred

1: Rising edge trigger request occurred

This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.

18.6.5 EXTI falling edge pending register (EXTI_FPR1)

Address offset: 0x010

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF16
															rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FPIF15	FPIF14	FPIF13	FPIF12	FPIF11	FPIF10	FPIF9	FPIF8	FPIF7	FPIF6	FPIF5	FPIF4	FPIF3	FPIF2	FPIF1	FPIF0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **FPIF[16:0]:** configurable event inputs x falling edge pending bit (x =0 to 16)

When EXTI_SECCFGR.SECx is disabled, FPIFx can be accessed with nonsecure and secure access.

When EXTI_SECCFGR.SECx is enabled, FPIFx can only be accessed with secure access. Nonsecure write to this FPIFx is discarded, nonsecure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, FPIFx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, FPIFx can only be accessed with privileged access. Unprivileged write to this FPIFx is discarded, unprivileged read returns 0.

0: No falling edge trigger request occurred

1: Falling edge trigger request occurred

This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing 1 to it.

18.6.6 EXTI security configuration register (EXTI_SECCFGR1)

Address offset: 0x014

Reset value: 0x0000 0000

This register provides write access security, a nonsecure write access is ignored and causes the generation of an illegal access event. A nonsecure read returns the register data.

Contains only register bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **SEC[16:0]:** Security enable on event input x (x = 0 to 16)
When EXTI_PRIVCFGR.PRIVx is disabled, SECx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, SECx can only be written with privileged access. Unprivileged write to this SECx is discarded.
0: event security disabled (nonsecure)
1: event security enabled (secure)

18.6.7 EXTI privilege configuration register (EXTI_PRIVCFGR1)

Address offset: 0x018

Reset value: 0x0000 0000

This register provides privileged write access protection. An unprivileged read returns the register data.

Contains only register bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **PRIV[16:0]:** Security enable on event input x (x = 0 to 16)
When EXTI_SECCFGR.SECx is disabled, PRIVx can be accessed with secure and nonsecure access.
When EXTI_SECCFGR.SECx is enabled, PRIVx can only be written with secure access. Nonsecure write to this PRIVx is discarded.
0: event privilege disabled (unprivileged)
1: event privilege enabled (privileged)

18.6.8 EXTI external interrupt selection register (EXTI_EXTICR1)

Address offset: 0x060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI3[7:0]								EXTI2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI1[7:0]								EXTI0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI3[7:0]:** EXTI3 GPIO port selection

These bits are written by software to select the source input for EXTI3 external interrupt.x
When EXTI_SECCFGR.SEC3 is disabled, EXTI3 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SEC3 is enabled, EXTI3 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIV3 is disabled, EXTI3 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV3 is enabled, EXTI3 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x01: PB3 pin
0x07: PH3 pin
Others: reserved

Bits 23:16 **EXTI2[7:0]:** EXTI2 GPIO port selection

These bits are written by software to select the source input for EXTI2 external interrupt.
When EXTI_SECCFGR.SEC2 is disabled, EXTI2 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SEC2 is enabled, EXTI2 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIV2 is disabled, EXTI2 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV2 is enabled, EXTI2 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA2 pin
0x01: PB2 pin
Others: reserved

Bits 15:8 **EXTI1[7:0]:** EXTI1 GPIO port selection

These bits are written by software to select the source input for EXTI1 external interrupt.
When EXTI_SECCFGR.SEC1 is disabled, EXTI1 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SEC1 is enabled, EXTI1 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIV1 is disabled, EXTI1 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV1 is enabled, EXTI1 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA1 pin
Others: reserved

Bits 7:0 **EXTI0[7:0]:** EXTI0 GPIO port selection

These bits are written by software to select the source input for EXTI0 external interrupt.
When EXTI_SECCFGR.SEC0 is disabled, EXTI0 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SEC0 is enabled, EXTI0 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIV0 is disabled, EXTI0 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV0 is enabled, EXTI0 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA0 pin
Others: reserved

Note: This field is reserved on STM32WBA25xx devices

18.6.9 EXTI external interrupt selection register (EXTI_EXTICR2)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI7[7:0]								EXTI6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI5[7:0]								EXTI4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI7[7:0]:** EXTI7 GPIO port selection

These bits are written by software to select the source input for EXTI7 external interrupt.
When EXTI_SECCFGR.SEC7 is disabled, EXTI7 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SEC7 is enabled, EXTI7 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIV7 is disabled, EXTI7 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV7 is enabled, EXTI7 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA7 pin
Others: reserved

Bits 23:16 **EXTI6[7:0]:** EXTI6 GPIO port selection

These bits are written by software to select the source input for EXTI6 external interrupt.
When EXTI_SECCFGR.SEC6 is disabled, EXTI6 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SEC6 is enabled, EXTI6 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIV6 is disabled, EXTI6 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV6 is enabled, EXTI6 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA6 pin
0x01: PB6 pin
Others: reserved

Bits 15:8 **EXTI5[7:0]:** EXTI5 GPIO port selection

These bits are written by software to select the source input for EXTI5 external interrupt.
When EXTI_SECCFGR.SEC5 is disabled, EXTI5 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SEC5 is enabled, EXTI5 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIV5 is disabled, EXTI5 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV5 is enabled, EXTI5 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA5 pin
0x01: PB5 pin
Others: reserved

Bits 7:0 **EXTI4[7:0]:** EXTI4 GPIO port selection

These bits are written by software to select the source input for EXTI4 external interrupt.
When EXTI_SECCFGR.SEC4 is disabled, EXTI4 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SEC4 is enabled, EXTI4 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIV4 is disabled, EXTI4 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV4 is enabled, EXTI4 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x01: PB4 pin
Others: reserved

18.6.10 EXTI external interrupt selection register (EXTI_EXTICR3)

Address offset: 0x068

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI11[7:0]								EXTI10[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI9[7:0]								EXTI8[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI11[7:0]:** EXTI11 GPIO port selection

These bits are written by software to select the source input for EXTI11 external interrupt.
When EXTI_SECCFGR.SEC11 is disabled, EXTI11 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SEC11 is enabled, EXTI11 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIV11 is disabled, EXTI11 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV11 is enabled, EXTI11 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA11 pin
Others: reserved

Bits 23:16 **EXTI10[7:0]:** EXTI10 GPIO port selection

These bits are written by software to select the source input for EXTI10 external interrupt.
When EXTI_SECCFGR.SEC10 is disabled, EXTI10 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SEC10 is enabled, EXTI10 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIV10 is disabled, EXTI10 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV10 is enabled, EXTI10 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA10 pin
Others: reserved

Bits 15:8 **EXTI9[7:0]:** EXTI9 GPIO port selection

These bits are written by software to select the source input for EXTI9 external interrupt.
When EXTI_SECCFGR.SEC9 is disabled, EXTI9 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SEC9 is enabled, EXTI9 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIV9 is disabled, EXTI9 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV9 is enabled, EXTI9 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA9 pin
0x01: PB9 pin
Others: reserved

Bits 7:0 **EXTI8[7:0]:** EXTI8 GPIO port selection

These bits are written by software to select the source input for EXTI8 external interrupt.
When EXTI_SECCFGR.SEC8 is disabled, EXTI8 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SEC8 is enabled, EXTI8 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIV8 is disabled, EXTI8 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV8 is enabled, EXTI8 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA8 pin
0x01: PB8 pin
Others: reserved

18.6.11 EXTI external interrupt selection register (EXTI_EXTICR4)

Address offset: 0x06C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI15[7:0]								EXTI14[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI13[7:0]								EXTI12[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI15[7:0]:** EXTI15 GPIO port selection

These bits are written by software to select the source input for EXTI15 external interrupt.
When EXTI_SECCFGR.SEC15 is disabled, EXTI15 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SEC15 is enabled, EXTI15 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIV15 is disabled, EXTI15 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV15 is enabled, EXTI15 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA15 pin
0x01: PB15 pin
0x02: PC15 pin
Others: reserved

Bits 23:16 **EXTI14[7:0]:** EXTI14 GPIO port selection

These bits are written by software to select the source input for EXTI14 external interrupt.
When EXTI_SECCFGR.SEC14 is disabled, EXTI14 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SEC14 is enabled, EXTI14 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIV14 is disabled, EXTI14 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV14 is enabled, EXTI14 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA14 pin
0x02: PC14 pin
Others: reserved

Bits 15:8 **EXTI13[7:0]:** EXTI13 GPIO port selection

These bits are written by software to select the source input for EXTI13 external interrupt.
When EXTI_SECCFGR.SEC13 is disabled, EXTI13 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SEC13 is enabled, EXTI13 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIV13 is disabled, EXTI13 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV13 is enabled, EXTI13 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA13 pin
0x02: PC13 pin
Others: reserved

Bits 7:0 **EXTI12[7:0]:** EXTI0 GPIO port selection

These bits are written by software to select the source input for EXTI12 external interrupt.
When EXTI_SECCFGR.SEC12 is disabled, EXTI12 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SEC12 is enabled, EXTI12 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIV12 is disabled, EXTI12 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV12 is enabled, EXTI12 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA12 pin
0x01: PB12 pin
Others: reserved

18.6.12 EXTI lock register (EXTI_LOCKR)

Address offset: 0x070

Reset value: 0x0000 0000

This register provides write access security: a nonsecure write access is ignored and a read access returns zero data, and both generates an illegal access event.

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

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **LOCK**: Global security and privilege configuration registers (EXTI_SECCFGR and EXTI_PRIVCFGR) lock

This bit is written once after reset.
0: security and privilege configuration open, can be modified.
1: security and privilege configuration locked, can no longer be modified.

18.6.13 EXTI CPU wake-up with interrupt mask register (EXTI_IMR1)

Address offset: 0x080

Reset value: 0x0000 0000

Contains register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **IM[16:0]:** CPU wake-up with interrupt mask on event input x [1] (x = 0 to 16)
When EXTI_SECCFGR.SECx is disabled, IMx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, IMx can only be accessed with secure access. Nonsecure write to this bit is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, IMx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, IMx can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0: Wake-up with interrupt request from input event x is masked.
1: Wake-up with interrupt request from input event x is unmasked.

1. The reset value for configurable event inputs is set to 0 in order to disable the interrupt by default.

18.6.14 EXTI CPU wake-up with event mask register (EXTI_EMR1)

Address offset: 0x084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **EM[16:0]:** CPU wake-up with event generation mask on event input x (x = 0 to 16)
When EXTI_SECCFGR.SECx is disabled, EMx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, EMx can only be accessed with secure access. Nonsecure write to this bit x is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, EMx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, EMx can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0: Wake-up with event generation from Line x is masked.
1: Wake-up with event generation from Line x is unmasked.

18.6.15 EXTI register map

Table 147. EXTI register map and reset values

Offset	Register	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
0x000	**EXTI_RTSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT16	RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	**EXTI_FTSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT16	FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	**EXTI_SWIER1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI16	SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	**EXTI_RPR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF16	RPIF15	RPIF14	RPIF13	RPIF12	RPIF11	RPIF10	RPIF9	RPIF8	RPIF7	RPIF6	RPIF5	RPIF4	RPIF3	RPIF2	RPIF1	RPIF0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	**EXTI_FPR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF16	FPIF15	FPIF14	FPIF13	FPIF12	FPIF11	FPIF10	FPIF9	FPIF8	FPIF7	FPIF6	FPIF5	FPIF4	FPIF3	FPIF2	FPIF1	FPIF0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	**EXTI_ SECCFG1R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC16	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	**EXTI_ PRIVCFG1R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C to 0x05C	Reserved	Reserved																															
0x060	**EXTI_EXTICR1**	EXTI3[7:0]								EXTI2[7:0]								EXTI1[7:0]								EXTI0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	**EXTI_EXTICR2**	EXTI7[7:0]								EXTI6[7:0]								EXTI5[7:0]								EXTI4[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	**EXTI_EXTICR3**	EXTI11[7:0]								EXTI10[7:0]								EXTI9[7:0]								EXTI8[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x06C	**EXTI_EXTICR4**	EXTI15[7:0]								EXTI14[7:0]								EXTI13[7:0]								EXTI2[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	**EXTI_LOCKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK
	Reset value																																0
0x074 to 0x07C	Reserved	Reserved																															
0x080	**EXTI_IMR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM16	IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 147. EXTI register map and reset values (continued)

Offset	Register	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
0x084	**EXTI_EMR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM16	EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088 to 0x3FC	Reserved	Reserved																															

Refer to *Section 2.3.2: Memory map and register boundary addresses* for the register boundary addresses.

19 Cyclic redundancy check calculation unit (CRC)

19.1 CRC introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location.

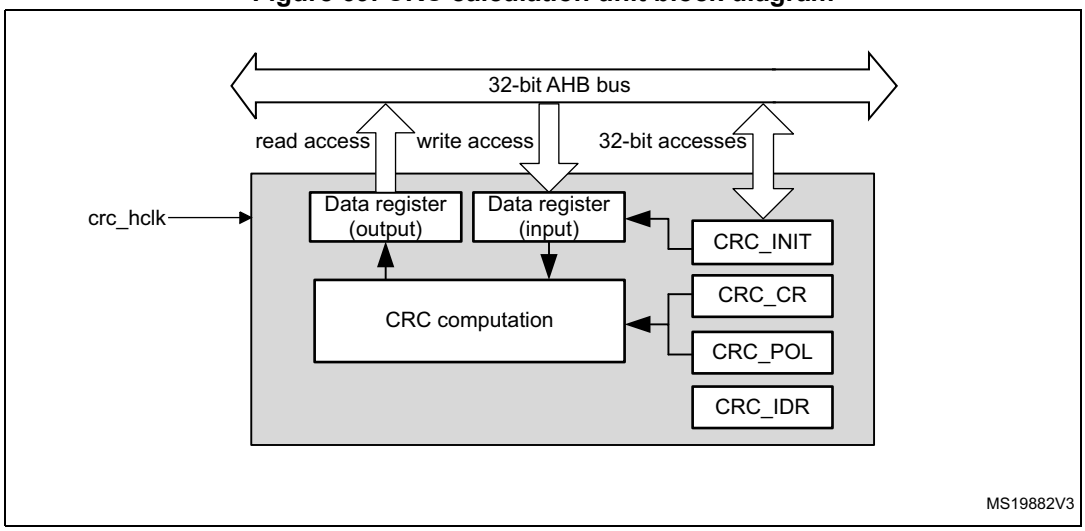
19.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-,16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility option on I/O data
- Accessed through AHB slave peripheral by 32-bit words only, with the exception of CRC_DR register that can be accessed by words, right-aligned half-words and right-aligned bytes

19.3 CRC functional description

19.3.1 CRC block diagram

Figure 69. CRC calculation unit block diagram



19.3.2 CRC internal signals

Table 148. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

19.3.3 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. For the other registers only 32-bit accesses are allowed.

The duration of the computation depends on input data width:

- 4 AHB clock cycles for 32 bits
- 2 AHB clock cycles for 16 bits
- 1 AHB clock cycles for 8 bits

An input buffer allows a second data to be immediately written without waiting for any wait-states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register.

For example, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x58D43CB2 with bit-reversal done by byte
- 0xD458B23C with bit-reversal done by half-word
- 0xB23CD458 with bit-reversal done on the full word

The output data can also be reversed by setting the REV_OUT bit in the CRC_CR register.

The operation is done at bit level. For example, 0x11223344 output data are converted to 0x22CC4488.

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC 7-, 8-, 16-, and 32-bit calculations. It is not affected by the RESET bit in the CRC_CR register.

Polynomial programmability

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7-, 8-, 16-, or 32-bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

Note: The type of an even polynomial is $X+X^2+..+X^n$, while the type of an odd polynomial is $1+X+X^2+..+X^n$.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size can not be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.

19.4 CRC registers

The CRC_DR register can be accessed by words, right-aligned half-words and right-aligned bytes. For the other registers only 32-bit accesses are allowed.

19.4.1 CRC data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

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

Bits 31:0 **DR[31:0]:** Data register bits

This register is used to write new data to the CRC calculator.

It holds the previous CRC calculation result when it is read.

If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

19.4.2 CRC independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bits 31:0 **IDR[31:0]**: General-purpose 32-bit data register bits

These bits can be used as a temporary storage location for four bytes.

This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register

19.4.3 CRC control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
								rw	rw	rw	rw	rw			rs

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **REV_OUT**: Reverse output data

This bit controls the reversal of the bit order of the output data.
0: Bit order not affected
1: Bit-reversed output format

Bits 6:5 **REV_IN[1:0]**: Reverse input data

This bitfield controls the reversal of the bit order of the input data
00: Bit order not affected
01: Bit reversal done by byte
10: Bit reversal done by half-word
11: Bit reversal done by word

Bits 4:3 **POLYSIZE[1:0]**: Polynomial size

These bits control the size of the polynomial.
00: 32 bit polynomial
01: 16 bit polynomial
10: 8 bit polynomial
11: 7 bit polynomial

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RESET**: RESET bit

This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware.

19.4.4 CRC initial value (CRC_INIT)

Address offset: 0x10

Reset value: 0xFFFF FFFF

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

Bits 31:0 **CRC_INIT[31:0]**: Programmable initial CRC value

This register is used to write the CRC initial value.

19.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14

Reset value: 0x04C1 1DB7

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

Bits 31:0 **POL[31:0]**: Programmable polynomial

This register is used to write the coefficients of the polynomial to be used for CRC calculation.

If the polynomial size is less than 32 bits, the least significant bits have to be used to program the correct value.

19.4.6 CRC register map

Table 149. CRC register map and reset values

Offset	Register name	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
0x00	CRC_DR	DR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	CRC_IDR	IDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	CRC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
	Reset value																									0	0	0	0	0			0
0x10	CRC_INIT	CRC_INIT[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	CRC_POL	POL[31:0]																															
	Reset value	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	1

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

20 Extended-SPI interface (XSPI)

20.1 XSPI introduction

The XSPI supports external serial memories such as serial NAND and serial NOR flash memories, with the following functional modes:

- indirect mode: all the operations are performed using the XSPI registers to preset commands, addresses, data, and transfer parameters.
- memory-mapped mode: the external memory is memory mapped and it is seen by the system as if it was an internal memory, supporting read operations.

The XSPI supports the following protocol with associated frame format:

- the regular-command frame format with the command, address, alternate byte, dummy cycles, and data phase

20.2 XSPI main features

- Functional modes: indirect and memory-mapped
- Read support in memory-mapped mode
- Support for single, dual, quad, octal communication
- Dual-memory configuration, where eight bits can be sent/received simultaneously by accessing two quad memories in parallel
- SDR (single-data rate) and DTR (double-transfer rate) support
- Data strobe support
- Fully programmable opcode
- Fully programmable frame format
- Integrated FIFO for reception and transmission
- 8-, 16-, and 32-bit data accesses allowed
- DMA protocol support
- DMA channel for indirect mode operations
- Interrupt generation on FIFO threshold, timeout, operation complete, and access error
- XSPI interface with transaction acceptance limited to one: the interface accepts the next transfer on AHB bus only once the previous is completed on memory side.
- Dual chip select support (NCS1 and NCS2)
- Possibility to disable the automatic prefetch

20.3 XSPI implementation

Table 150. XSPI implementation

XSPI feature	XSPI1
Octal mode configuration (8-bit)	-
Quad mode configuration (4-bit)	X
16-bit mode configuration	-

Table 150. XSPI implementation (continued)

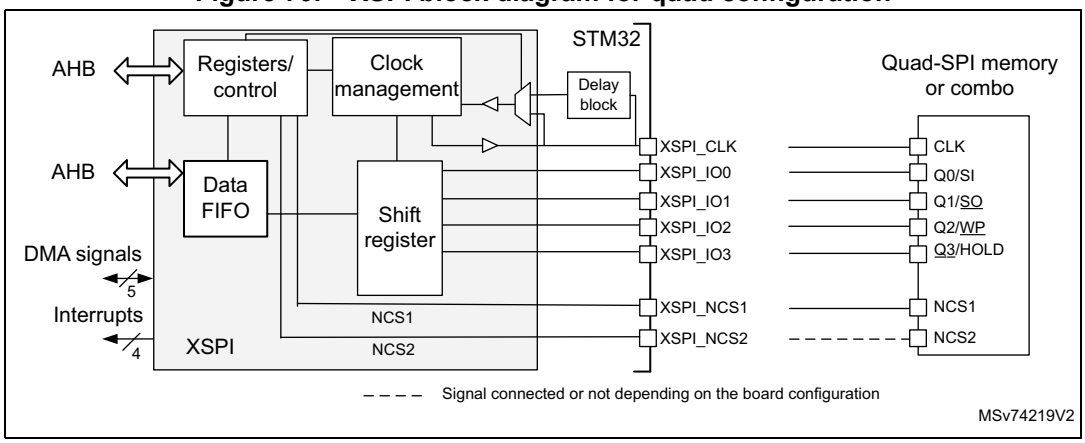
XSPI feature	XSPI1
HyperBus standard compliant	-
Xccela standard compliant	X
XSPI (JESD251C) standard compliant	X
AMBA® AHB compliant data interface	X
Dual AHB interface	X
Asynchronous AHB clock versus kernel clock	-
Functional modes: indirect, automatic status-polling, and memory-mapped(1)(2)	X
Read and write support in memory-mapped mode(2)	X
Dual-quad configuration	-
SDR (single-data rate) and DTR (double-transfer rate)	X
Data strobe (DS,DQS)	X
Fully programmable opcode	X
Fully programmable frame format	X
Integrated FIFO for reception and transmission	X
8-, 16-, and 32-bit data accesses	X
Interrupt on FIFO threshold, timeout, operation complete, and access error	X
Bus error bit	-
Address offset	-
Extended CSHT timeout	X
Memory-mapped write	-
Refresh counter	-
GPDMA interface	X
Dual chip select	-
Extended external memory	-
Prefetch disable	X
Prefetch hardware software	-

1. Automatic status polling mode is not supported.
2. Memory-mapped write are not supported.

20.4 XSPI functional description

20.4.1 XSPI block diagram

Figure 70. XSPI block diagram for quad configuration



20.4.2 XSPI pins and internal signals

Table 151. XSPI input/output pins

Pin name	Type	Description
XSPI_CLK	Output	XSPI clock
XSPI_IOn (n = 0 to 7)	Input/output	XSPI data pins(1)
XSPI_NCS1,2	Output	Chip select for the memory(2)
XSPI_DQS	Input/output	Data strobe/write mask signal from/to the memory

1. XSPI_IO[7:4] is not supported.
2. XSPI_NCS2 is not supported.

Caution: Use the same configuration (output speed, HSLV[a]) for all XSPI input/output pins to avoid any data corruption.

Table 152. XSPI internal signals

Signal name	Type	Description
xspi_hclk	Input	XSPI AHB clock
xspi_dma	N/A	DMA request signal
xspi_it	Output	Global interrupt line (see *Table 155* for the multiple sources of interrupt)

a. When applicable.

20.4.3 XSPI interface to memory modes

The XSPI supports the following protocol:

- regular-command protocol

The XSPI uses from 6 to 12 signals to interface with a memory, depending on the functional mode:

- NCS: chip-select (in the following sections, unless otherwise specified, NCS applies for NCS1 or NCS2. Only one of these two signals is active at a given moment in time)
- CLK: communication clock
- DQS: data strobe used only in regular-command protocol as input only
- IO[3:0]: data bus LSB
- IO[7:4]:
  - data bus MSB used in dual-quad and octal configurations
  - data bus can be used as possible remap for quad-SPI mode

20.4.4 XSPI regular-command protocol

When in regular-command protocol, the XSPI communicates with the external device using commands. Each command can include the following phases:

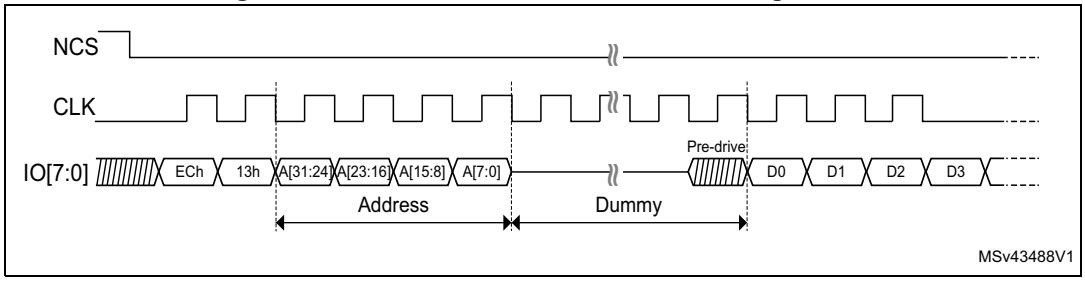
- Instruction phase
- Address phase
- Alternate-byte phase
- Dummy-cycle phase
- Data phase

Any of these phases can be configured to be skipped but, in case of single-phase command, the only use case supported is instruction-phase-only.

The NCS falls before the start of each command and rises again after each command finishes.

In memory-mapped mode, read operations are supported.

Figure 71. SDR read command in octal configuration



The specific regular-command protocol features are configured through the registers in the 0x0100-0x01FC offset range.

Instruction phase

During this phase, a 1- to 4-byte instruction is sent to the external device specifying the type of operation to be performed. The size of the instruction to be sent is configured

by ISIZE[1:0] in XSPI_CCR and the instruction is programmed in INSTRUCTION[31:0] of XSPI_IR.

The instruction phase can optionally send:

- 1 bit at a time (over IO0, SO signal in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using IMODE[2:0] of XSPI_CCR.

The instruction can be sent in DTR mode on each rising and falling edge of the clock, by setting IDTR in XSPI_CCR.

When IMODE[2:0] = 000 in XSPI_CCR, the instruction phase is skipped, and the command sequence starts with the address phase, if present.

In memory-mapped mode, the instruction used for the read operation and the instruction format are specified in XSPI_IR and XSPI_CCR.

Address phase

In the address phase, 1 to 4 bytes are sent to the external device, to indicate the address of the operation. The number of address bytes to be sent is configured by ADSIZE[1:0] in XSPI_CCR.

In indirect mode, the address bytes to be sent are specified by ADDRESS[31:0] in XSPI_AR. In memory-mapped mode, the address is given directly via the AHB (from any master in the system).

The address phase can send:

- 1 bit at a time (over IO0, SO signal in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using ADMODE[2:0] in XSPI_CCR.

The address can be sent in DTR mode (on each rising and falling edge of the clock) setting ADDTR in XSPI_CCR.

When ADMODE[2:0] = 000, the address phase is skipped and the command sequence proceeds directly to the next phase, if any.

In memory-mapped mode, the address format for the read operation is specified in XSPI_CCR.

Alternate-byte phase

In the alternate-byte phase, 1 to 4 bytes are sent to the external device, generally to control the mode of operation. The number of alternate bytes to be sent is configured by ABSIZE[1:0] in XSPI_CCR. The bytes to be sent are specified in XSPI_ABR.

The alternate-byte phase can send:

- 1 bit at a time (over IO0, SO signal in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using ABMODE[2:0] in XSPI_CCR.

The alternate bytes can be sent in DTR mode (on each rising and falling edge of the clock) setting ABDTR in XSPI_CCR.

When ABMODE[2:0] = 000, the alternate-byte phase is skipped and the command sequence proceeds directly to the next phase, if any.

There may be times when only a single nibble needs to be sent during the alternate-byte phase rather than a full byte, such as when the dual-SPI mode is used and only two cycles are used for the alternate bytes.

In this case, the firmware can use the quad-SPI mode (ABMODE[2:0] = 011) and send a byte with bits 7 and 3 of ALTERNATE[31:0] set to 1 (keeping the IO3 line high), and bits 6 and 2 set to 0 (keeping the IO2 line low), in XSPI_IR.

The upper two bits of the nibble to be sent are then placed in bits 5:4 of ALTERNATE[31:0] while the lower two bits are placed in bits 1:0. For example, if the nibble 2 (0010) is to be sent over IO0/IO1, then ALTERNATE[31:0] must be set to 0x8A (1000_1010).

In memory-mapped mode, the alternate bytes used for read operation and the alternate byte format are specified in XSPI_ABR and XSPI_CCR.

Dummy-cycle phase (memory latency)

In the dummy-cycle phase, 1 to 31 cycles are given without any data being sent or received, in order to give the external device, the time to prepare for the data phase when higher clock frequencies are used. The number of cycles given during this phase is specified by DCYC[4:0] in XSPI_TCR. In both SDR and DTR modes, the duration is specified as a number of full CLK cycles.

When DCYC[4:0] = 00000, the dummy-cycle phase is skipped, and the command sequence proceeds directly to the data phase, if present.

In order to assure enough “turn-around” time for changing the data signals from the output mode to the input mode, there must be at least one dummy cycle when using the dual-, the quad-, or the octal-SPI mode, to receive data from the external device.

In memory-mapped mode, the dummy cycles for the read operation are specified in XSPI_TCR.

Data phase

During the data phase, any number of bytes can be sent to or received from the external device.

In indirect mode, the number of bytes to be sent/received is specified in XSPI_DLR. In this mode, the data to be sent to the external device must be written to XSPI_DR, while in indirect-read mode the data received from the external device is obtained by reading XSPI_DR.

In memory-mapped mode, the data read is received directly over the AHB to the Cortex core or to a DMA.

The data phase can send/receive:

- 1 bit at a time (over IO0/IO1 (SO/SI respectively) in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using DMODE[2:0] in XSPI_CCR.

The data can be sent or received in DTR mode (on each rising and falling edge of the clock) setting DDTR in XSPI_CCR.

When DMODE[2:0] = 000, the data phase is skipped, and the command sequence finishes immediately by raising the NCS. This configuration must be used only in indirect-write mode.

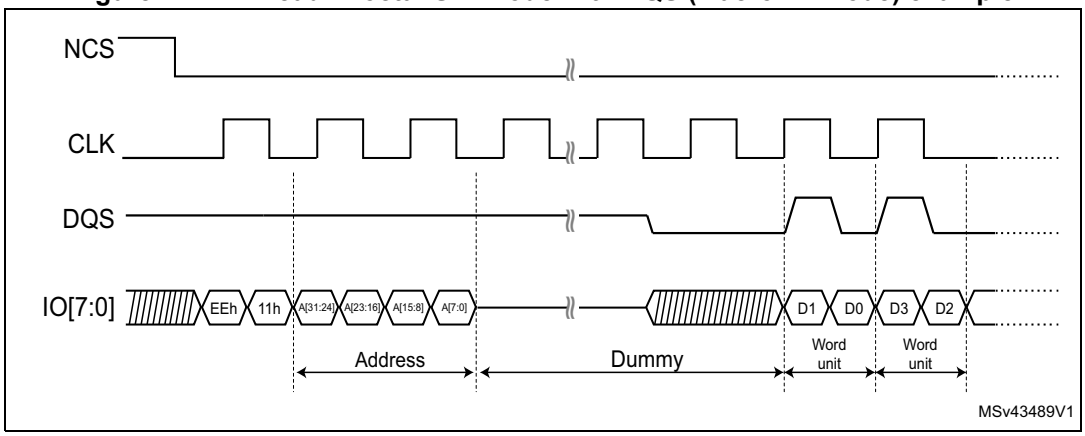
In memory-mapped mode, the data format for the read operation is specified in XSPI_CCR.

DQS use

The DQS signal can be used for data strobing during the read transactions when the device toggles the DQS aligned with the data.

The DQS management can be enabled by setting DQSE in XSPI_CCR.

Figure 72. DTR read in octal-SPI mode with DQS (Macronix mode) example



20.4.5 XSPI regular-command protocol signal interface

Single-SPI mode

The legacy SPI mode allows just a single bit to be sent/received serially. In this mode, the data is sent to the external device over the SO signal (Single-SPI Output) (whose I/Os are shared with IO0). The data received from the external device arrives via SI (Single-SPI Input) (whose I/Os are shared with IO1).

Compared to the SPI legacy mode, I0/SO and I1/SI are respectively equivalent to MOSI and MISO, having the XSPI generating the clock.

The different phases can each be configured separately to use this single-SPI mode by setting to 001 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase configured in single-SPI mode:

- IO0 (SO) is in output mode.
- IO1 (SI) is in input mode (high impedance).
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1 (to deactivate the “hold” function).
- IO4 to IO7 are in output mode and forced to 0.

This is the case even for the dummy phase if DMODE[2:0] = 001.

Dual-SPI mode

In dual-SPI mode, two bits are sent/received simultaneously over the IO0/IO1 signals.

The different phases can each be configured separately to use dual-SPI mode by setting to 010 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase configured in dual-SPI mode:

- IO0/IO1 are at high-impedance (input) during the data phase for the read operations, and outputs in all other cases.
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1.
- IO4 to IO7 are in output mode and forced to 0.

In the dummy phase, when DMODE[2:0] = 010, IO0 and IO1 are in a high-impedance state during read transactions, and are forced to either high or low levels during write transactions.

Quad-SPI mode

In quad-SPI mode, four bits are sent/received simultaneously over the IO0/IO1/IO2/IO3 signals.

The different phases can each be configured separately to use the quad-SPI mode by setting to 011 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase configured in quad-SPI mode:

- IO0 to IO3 are all at high-impedance (inputs) during the data phase for the read operations, and outputs in all other cases.
- IO4 to IO7 are in output mode and forced to 0.

In the dummy phase, when DMODE[2:0] = 011, IO0 to IO3 are in a high-impedance state during read transactions, and are forced to either high or low levels during write transactions.

Octal-SPI mode

In regular octal-SPI mode, the eight bits are sent/received simultaneously over the IO[0:7] signals.

The different phases can each be configured separately to use the octal-SPI mode by setting to 100 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase that is configured in octal-SPI mode, IO[0:7] are all at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase, when DMODE[2:0] = 100, IO[0:7] are in a high-impedance state during read transactions, and are forced to either high or low levels during write transactions.

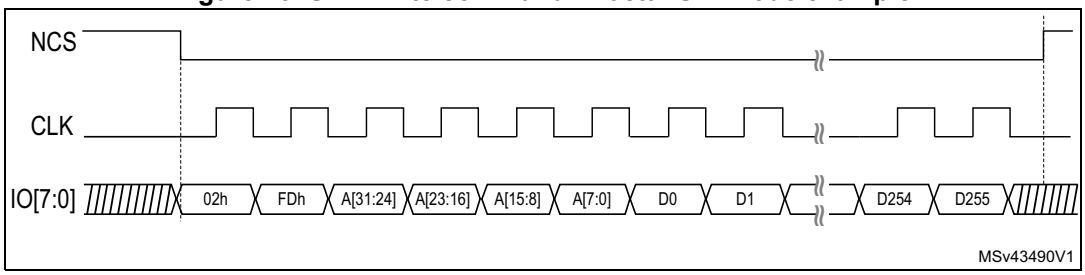
Single-data rate (SDR) mode

By default, all the phases operate in single-data rate (SDR) mode.

In SDR mode, when the XSPI drives the IO0/SO, IO1 to IO7 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the XSPI assumes that the external devices also send the data using CLK falling edge. By default (when SSHIFT = 0 in XSPI_TCR), the signals are sampled using the following (rising) edge of CLK.

Figure 73. SDR write command in octal-SPI mode example



Double-transfer rate (DTR) mode

Each of the instruction, address, alternate-byte, and data phases can be configured to operate in DTR mode setting IDTR, ADDTR, ABDTR, and DDTR in XSPI_CCR.

In memory-mapped mode, the DTR mode for each phase of the read operations is specified in XSPI_CCR.

In DTR mode, when the XSPI drives the IO0/SO and IO1 to IO7 signals in the instruction, address, and alternate-byte phases, a bit is sent or received on each of the falling and rising edges of CLK.

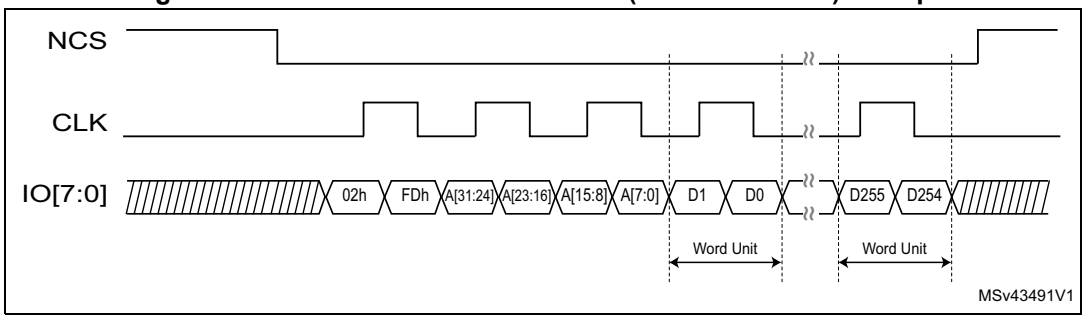
In DTR mode, when the XSPI drives the IO0 to IO7 signals in the data phases, a bit is sent or received on each of the falling and rising edges of CLK.

When receiving data in DTR mode, the XSPI assumes that the external devices also send the data using both CLK rising and falling edges. When DDTR = 1 in XSPI_CCR, the software must clear SSHIFT in XSPI_TCR. Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).

In DTR mode, it is recommended to set DHQC of XSPI_TCR, to shift the outputs by a quarter of cycle and avoid holding issues on the memory side.

Note: DHQC must not be set when the prescaler value is 0, as this action leads to unpredictable behavior.

Figure 74. DTR write in octal-SPI mode (Macronix mode) example



Dual-quad configuration

When DMM = 1 in XSPI_CR, the XSPI is in dual-memory configuration: if DMODE = 011, two external quad-SPI devices (device A and device B) are used in order to send/receive eight bits (or 16 bits in DTR mode) every cycle, effectively doubling the throughput.

Each device (A or B) uses the same CLK and NCS signals, but each has separate IO0 to IO3 signals.

The dual-quad configuration can be used in conjunction with the single-SPI, dual-SPI, and quad-SPI modes, as well as with either the SDR or DTR mode.

The device size, as specified by DEVSIZE[4:0] in XSPI_DCR1, must reflect the total external device capacity that is the double of the size of one individual component.

If address X is even, then the byte that the XSPI gives for address X is the byte at the address X/2 of device A, and the byte that the XSPI gives for address X + 1 is the byte at the address X/2 of device B. In other words, the bytes at even addresses are all stored in device A and the bytes at odd addresses are all stored in device B.

When reading the status registers of the devices in dual-quad configuration, twice as many bytes must be read compared to the same read in regular-command protocol: if each device gives eight valid bits after the instruction for fetching the status register, then the XSPI must be configured with a data length of 2 bytes (16 bits), and the XSPI receives one byte from each device.

If each device gives a status of 16 bits, then the XSPI must be configured to read 4 bytes to get all the status bits of both devices in dual-quad configuration. The least-significant byte of the result (in the data register) is the least-significant byte of device A status register. The next byte is the least-significant byte of device B status register. Then, the third byte of the data register is the device A second byte. The fourth byte is the device B second byte (if devices have 16-bit status registers).

An even number of bytes must always be accessed in dual-quad configuration. For this reason, bit 0 of DL[31:0] in XSPI_DLR is stuck at 1 when DMM = 1.

In dual-quad configuration, the behavior of device A interface signals is basically the same as in normal mode. Device B interface signals have exactly the same waveforms as device A ones during the instruction, address, alternate-byte, and dummy-cycle phases. In other words, each device always receives the same instruction and the same address.

Then, during the data phase, the AIOx and the BIOx buses both transfer data in parallel, but the data that is sent to (or received from) device A is distinct than the one from device B.

20.4.6 XSPI operating mode introduction

The XSPI has the following operating modes regardless of the low-level protocol used :

- indirect mode (read or write)
- memory-mapped mode

20.4.7 XSPI indirect mode

In indirect mode, the commands are started by writing to the XSPI registers, and data are transferred by writing or reading the data register, in a similar way to other communication peripherals.

When FMODE[1:0] = 00 in XSPI_CR, the XSPI is in indirect-write mode: bytes are sent to the external device during the data phase. Data are provided by writing to XSPI_DR.

When FMODE[1:0] = 01, the XSPI is in indirect-read mode: bytes are received from the external device during the data phase. Data are recovered by reading XSPI_DR.

In indirect mode, when the XSPI is configured in DTR mode over eight lanes with DQS disabled, the given starting address and the data length must be even.

Note: The XSPI_AR register must be updated even if the start address is the same as the start address of the previous indirect access.

The number of bytes to be read/written is specified in XSPI_DLR:

- If DL[31:0] = 0xFFFF FFFF, the data length is considered undefined and the XSPI simply continues to transfer data until it reaches the end of the external device (as defined by DEVSIZE). If no bytes are to be transferred, DMODE[2:0] must be set to 0 in XSPI_CCR.
- If DL[31:0] = 0xFFFF FFFFF and DEVSIZE[4:0] = 0x1F (its maximum value indicating at 4-Gbyte device), the transfers continue indefinitely, stopping only after an abort request or after the XSPI is disabled. After the last memory address is read (at address 0xFFFF FFFF), reading continues with address = 0x0000 0000.

When the programmed number of bytes to be transmitted or received is reached, the TCF bit is set in XSPI_SR, and an interrupt is generated if TCIE = 1 in XSPI_CR. In the case of an undefined number of data, TCF is set when the limit of the external SPI memory is reached, according to the device size defined in XSPI_DCR1.

Triggering the start of a transfer in regular-command protocol

Depending on the XSPI configuration, there are three different ways to trigger the start of a transfer in indirect mode when using the regular-command protocol. In general, the start of transfer is triggered as soon as the software gives the last information that is necessary for

the command. More specifically in indirect mode, a transfer starts when one of the following sequence of events occurs:

- if no address is necessary (ADMODE[2:0] = 000) and if no data need to be provided by the software (FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to INSTRUCTION[31:0] in XSPI_IR
- if an address is necessary (when ADMODE[2:0] ≠ 000) and if no data need to be provided by the software (when FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to ADDRESS[31:0] in XSPI_AR
- if data need to be provided by the software (when FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), and at the moment when a write is performed to DATA[31:0] in XSPI_DR

A write to XSPI_ABR never triggers the communication start. If alternate bytes are required, they must have been programmed before.

As soon as a command is started, the BUSY bit is automatically set in XSPI_SR.

FIFO and data management

Data in indirect mode pass through a 32-byte FIFO that is internal to the XSPI. FLEVEL in XSPI_SR indicates how many bytes are currently being held in the FIFO.

AHB burst transactions are supported. Data of the burst are successively written in XSPI_DR, and immediately transferred in the internal FIFO.

In indirect-write mode (FMODE[1:0] = 00), the software adds data to the FIFO when it writes in XSPI_DR. A word write adds 4 bytes to the FIFO, a half-word write adds 2 bytes, and a byte write adds only 1 byte. If the software adds too many bytes to the FIFO (more than indicated in DL[31:0]), the extra bytes are flushed from the FIFO at the end of the write operation (when TCF is set).

The byte/half-word accesses to XSPI_DR must be done only to the least significant byte/halfword of the 32-bit register.

FTHRES is used to define a FIFO threshold after which point the FIFO threshold flag, FTF, gets set. In indirect-read mode, FTF is set when the number of valid bytes to be read from the FIFO is above the threshold. FTF is also set if there is any data left in the FIFO after the last byte is read from the external device, regardless of FTHRES setting. In indirect-write mode, the FTF is set when the number of empty bytes in the FIFO is above the threshold.

If FTIE = 1, there is an interrupt when the FTF is set. If DMAEN = 1, a DMA transfer is initiated when the FTF is set. The FTF is cleared by hardware as soon as the threshold condition is no longer true (after enough data has been transferred by the CPU or DMA).

The last data read in RX FIFO remains valid as long as there is no request for the next line. This means that, when the application reads several times in a row at the same location, the data is provided from the RX FIFO and not read again from the distant memory.

20.4.8 XSPI memory-mapped mode

When configured in memory-mapped mode, the external SPI device is seen as an internal memory.

Note: No more than 256 Mbytes can be addressed even if the external device capacity is larger.

If an access is made to an address outside of the range defined by DEVSIZE[4:0] but still within the 256-Mbyte range, then an AHB error is given. The effect of this error depends on the AHB master that attempted the access:

- If it is the Cortex CPU, a hard-fault interrupt is generated.
- If it is a DMA, a DMA transfer error is generated, and the corresponding DMA channel is automatically disabled.

Byte, half-word, and word access types are all supported.

A support for execute in place (XIP) operation is implemented, where the XSPI continues to load the bytes to the addresses following the most recent access. If subsequent accesses are continuous to the bytes that follow, then these operations end up quickly since their results were prefetched.

By default, the XSPI never stops its prefetch operation. It either keeps the previous read operation active with the NCS maintained low or it relaunches a new transfer, even if no access to the external device occurs for a long time. This prefetch feature can be disabled by setting the NOPREF bit in XSPI_DCR1. Disabling the prefetch can be beneficial for performance when the transfer requests are short and jumping frequently between non-consecutive addresses.

Since external devices tend to consume more power when the NCS is held low, the application may want to activate the timeout counter (TCEN = 1 in XSPI_CR): the NCS is released after a period defined by TIMEOUT[15:0] in XSPI_LPTR, when x cycles have elapsed without access since the clock is inactive.

BUSY goes high as soon as the first memory-mapped access occurs. Because of the prefetch operations, BUSY does not fall until there is an abort, or the peripheral is disabled.

It is recommended to add a synchronization barrier between the end of the controller registers configuration and the first memory-mapped access to the external memory when the controller is configured in memory-mapped mode.

The controller can apply an automatic address offset to simplify memory area code switching, such as during a firmware update (if the address offset feature is available in the product implementation table). To enable this feature, set the ADOFFEN bit in the XSPI_CR register and configure the offset value in the ADOFF[4:0] bitfield of the XSPI_DCR1 register. The offset is modulo 256 bytes.

20.4.9 XSPI configuration introduction

The XSPI configuration is done in three steps:

1. XSPI system configuration
2. XSPI device configuration
3. XSPI mode configuration

20.4.10 XSPI system configuration

The XSPI is configured using XSPI_CR. The user must program:

- the functional mode with FMODE[1:0]
- the FIFO level with FTHRES
- the DMA use with DMAEN
- the timeout counter use with TCEN

- the dual-memory configuration, if needed, with DMM

In case of an interrupt use, the respective enable bit can also be set during this phase.

If the timeout counter is used, the timeout value is programmed in XSPI_LPTR.

The DMA channel must not be enabled during the XSPI configuration: it must be enabled only when the operation is fully configured, to avoid any unexpected request generation.

The DMA and XSPI must be configured in a coherent manner regarding data length: FTHRES value must reflect the DMA burst size.

20.4.11 XSPI device configuration

The parameters related to the external device targeted are configured through XSPI_DCR1 and XSPI_DCR2.The user must program:

- the device size with DEVSIZE[4:0]
- the chip-select minimum high time with CSHT[5:0]
- the clock mode with FRCK and CKMODE
- the associated delay block DLYB connected externally to the peripheral
- the device frequency with PRESCALER[7:0]

DEVSIZE[4:0] defines the size of external memory using the following formula:

$$\text{Number of bytes in the device} = 2^{[\text{DEVSIZE}+1]}$$

where DEVSIZE+1 is the number of address bits required to address the external device. The external device capacity can go up to 4 Gbytes (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

If DMM = 1, DEVSIZE[4:0] must reflect the total capacity of the two devices together considering the above formula (DEVSIZE[4:0] value is so equal to one of the two memory capacities).

When the XSPI executes two commands, one immediately after the other, it raises the chip-select signal (NCS) high between the two commands for at least one CLK cycle by default.

If the external device requires more time between commands, CSHT[5:0] can be used to specify the minimum number of CLK cycles for which the NCS must remain high.

CKMODE indicates the level that the CLK takes between commands (when NCS = 1).

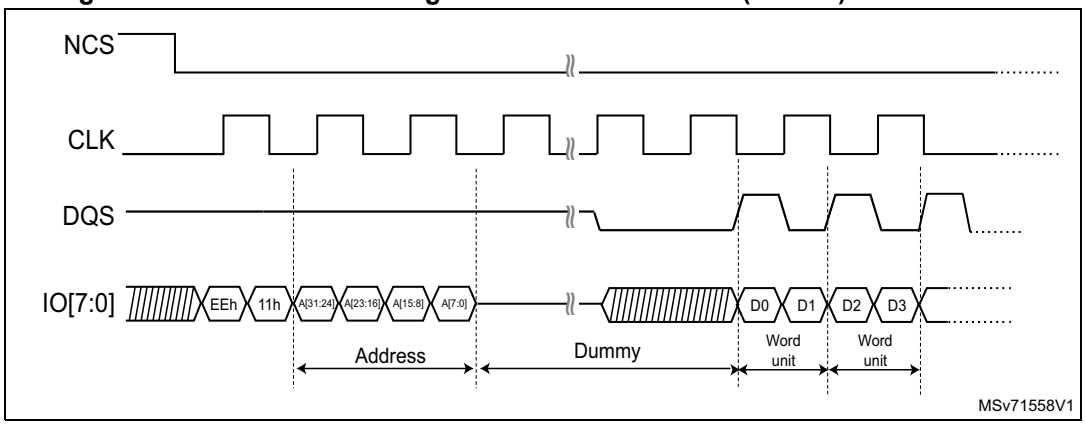
Memory types

External memory providers may present some architecture and slight data formatting differences between them. The bitfield MTYP[2:0] into the XSPI_CR register allows targeting the right controller configuration depending on the associated memory type selected in the application. This is the responsibility of the software developer to align the controller configuration to fit with the targeted memory type.

The memory types are grouped in a such way:

- D0/D1 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 000. For instance, Micron is using such data ordering. In this configuration, the DQS is sent with a polarity inverted respect to the clock polarity.

Figure 75. D0/D1 data ordering in octal-SPI DTR mode (Micron) - Read access



- D1/D0 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 001. For instance, Macronix is using this reverse data ordering in its Octaflash portfolio (this configuration is not adapted to its OctaRAM™ memories). DQS is keeping the same polarity as the clock when reading data from the memory. Refer to *Figure 72: DTR read in octal-SPI mode with DQS (Macronix mode) example*.
- D1/D0 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 011 with specific address phase built with row and column to fit with Macronix OctaRAM™ memories requirement (refer to *Table 153: OctaRAM command address bit assignment (based on 64 Mb OctaRAM)*. This is the controller which translates internally the targeted address provided by the software in row/column address formatting to sent to the memory. DQS is keeping the same polarity as the clock one when reading data from the memory.

Figure 76. OctaRAM read operation with reverse data ordering D1/D0

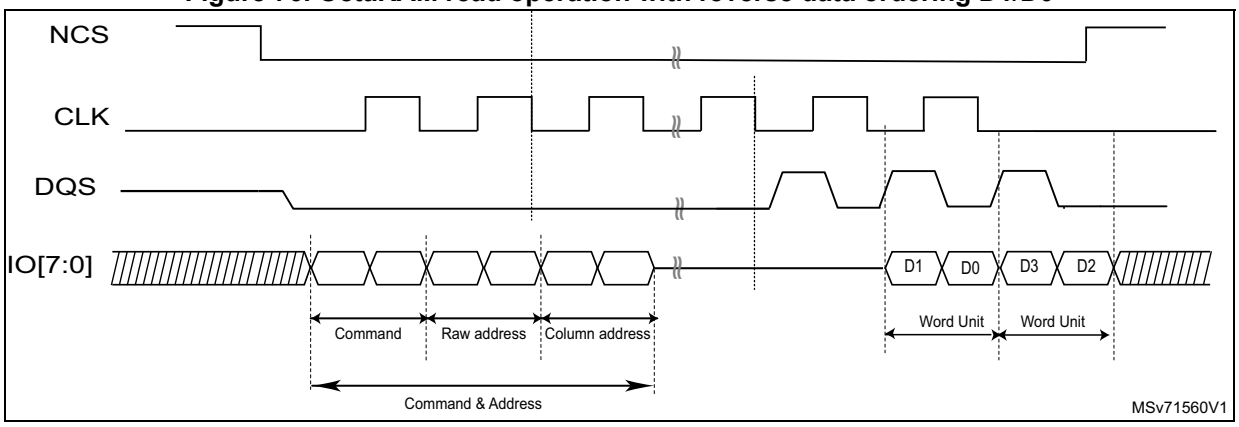


Table 153. OctaRAM command address bit assignment (based on 64 Mb[1] OctaRAM)

Clock	1st clock	2nd clock		3rd clock	
Function	Command	Row address		Column address	
SIO[7]	Command	Reserved	RA7	CA9	Reserved
SIO[6]		Reserved	RA6	CA8	Reserved
SIO[5]		Reserved	RA5	CA7	Reserved
SIO[4]		RA12	RA4	CA6	Reserved
SIO[3]		RA11	RA3	CA5	CA3
SIO[2]		RA10	RA2	CA4	CA2
SIO[1]		RA9	RA1	Reserved	CA1
SIO[0]		RA8	RA0	Reserved	CA0[2]

1. Example of 64 Mb OctaRAM address assignment:
Row Address [RA12:RA0]: 8K. Column Address [CA9:CA0]: 1K. 64 Mb density = 8K x 1K x 8 bits
2. Column address A0 must be always 0.

- Standard mode. It is the mode to use whenever the targeted memory is not corresponding to any other configurations described in this section. MTYP[2:0] = 010 for this standard mode.

20.4.12 XSPI regular-command mode configuration

Indirect mode configuration

When FMODE[1:0] = 00, the indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the indirect-read mode is selected, and data can be read from the external device.

When the XSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in XSPI_DLR.
2. Specify the frame timing in XSPI_TCR.
3. Specify the frame format in XSPI_CCR.
4. Specify the instruction in XSPI_IR.
5. Specify the optional alternate byte to be sent right after the address phase in XSPI_ABR.
6. Specify the targeted address in XSPI_AR.
7. Enable the DMA channel if needed.
8. Read/write the data from/to the FIFO through XSPI_DR (if no DMA usage).

If neither the address register (XSPI_AR) nor the data register (XSPI_DR) need to be updated for a particular command, then the command sequence starts as soon as XSPI_IR is written. This is the case when both ADMODE[2:0] and DMODE[2:0] equal 000, or if just ADMODE[2:0] = 000 when in indirect-read mode (FMODE[1:0] = 01).

When an address is required (ADMODE[2:0] ≠ 000) and the data register does not need to be written (FMODE[1:0] = 01 or DMODE[2:0] = 000), the command sequence starts as soon as the address is updated with a write to XSPI_AR.

In case of data transmission (FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), the communication start is triggered by a write in the FIFO through XSPI_DR.

Memory-mapped mode configuration

In memory-mapped mode, the external device is seen as an internal memory but with some latency during accesses. Read operations are allowed to the external device in this mode.

The memory-mapped mode is entered by setting FMODE[1:0] = 11 in XSPI_CR.

The programmed instruction and frame are sent when an AHB master accesses the memory-mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to XSPI_DR in this mode returns zero.

The data length register (XSPI_DLR) has no meaning in memory-mapped mode.

When the XSPI is used in memory-mapped mode, the frames are constructed in the following way:

1. Specify the frame timing in XSPI_TCR for read operation.
2. Specify the frame format in XSPI_CCR for read operation.
3. Specify the instruction in XSPI_IR.
4. Specify the optional alternate byte to be sent right after the address phase in XSPI_ABR for read operation.

All configuration operations must be completed (ensured by checking BUSY = 0) before the first access to the memory area: any register write operation when BUSY = 1 has no effect and is not signaled with an error response. On the first access, the XSPI becomes busy, and no further configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit or to abort by setting the ABORT bit.

XSPI delayed data sampling when no DQS is used

By default, when no DQS is used, the XSPI samples the data driven by the external device one half of a CLK cycle after the external device drives the signal.

In case of any external signal delays, it may be useful to sample the data later. Using SSHIFT in XSPI_TCR, the sampling of the data can be shifted by half of a CLK cycle.

The firmware must clear SSHIFT when the data phase is configured in DTR mode (DDTR = 1).

XSPI delayed data sampling when DQS is used

When external DQS is used as a sampling clock, it is shifted precisely by one quarter of the SPI clock cycle, for all frequencies above freq_min, to compensate the data propagation delay in the “high-speed interface” when the product embeds one.

This shift is performed by an external delay block (DLYB) located outside the XSPI. This external delay block has to be calibrated (by enabling FRCK) before starting the first transfer, and recalibrated regularly. The control of this feature depends on the device implementation (see the delay block (DLYB) section in this document for more details about DLYB and calibration procedure, and the product datasheet for freq_min value).

In configurations where delay does not need to be compensated, the external delay block can be bypassed by setting DLYBYP in XSPI_DCR1.

20.4.13 XSPI error management

The following errors set the TEF flag in XSPI_SR and generates an interrupt if enabled (TEIE = 1 in XSPI_CR):

- in indirect mode, when a wrong address has been programmed in XSPI_AR (according to the device size defined by DEVSIZE[4:0]).
- in indirect mode, if the address plus the data length exceed the device size: TEF is set as soon as the access is triggered.

In memory-mapped mode, the XSPI generates an AHB slave error in the following situations, setting the BERRF flag in XSPI_SR and generating an interrupt if enabled (BERRIE = 1 in XSPI_CR) (if the Bus error bit feature is available in the product implementation table, otherwise the Slave bus error response is generated at system level as transaction response with no dedicated bit into the controller status register):

- The memory-mapped mode is disabled and an AHB read or write request occurs.
- A read (including the address offset if ADOFFEN bit is set) address exceeds the size of the external memory.
- An abort is received while a read burst is ongoing when the abort condition is present at the moment of transfer phases preceding the data phase (In incremental). To avoid the error, refer to *Section 20.4.14: XSPI BUSY and ABORT*.
- The XSPI is disabled while a read burst is requested.
- A read request is received while DMODE[2:0] = 000 (no data phase).
- Bit DMM is set in XSPI_CR (dual-memory configuration) and octal mode is set.
- A write request is received.

20.4.14 XSPI BUSY and ABORT

Once the XSPI starts an operation with the external device, BUSY is automatically set in XSPI_SR.

In indirect mode, BUSY is reset once the XSPI has completed the requested command sequence and the FIFO is empty.

After the first access in memory-mapped mode, BUSY goes low only on an abort.

Any operation can be aborted by setting ABORT in XSPI_CR. Once the abort is completed, BUSY and ABORT are automatically reset, and the FIFO is flushed.

Before setting ABORT, the software must ensure that all the current transactions are finished using the synchronization barriers. When DMA is enabled to handle the data read or write operations in XSPI_DR, it is recommended to disable the DMA channel before aborting the SPI.

Note: Some devices may misbehave if a write operation to a status register is aborted.

20.4.15 XSPI reconfiguration or deactivation

Before any XSPI reconfiguration, the software must ensure that all the transactions are completed:

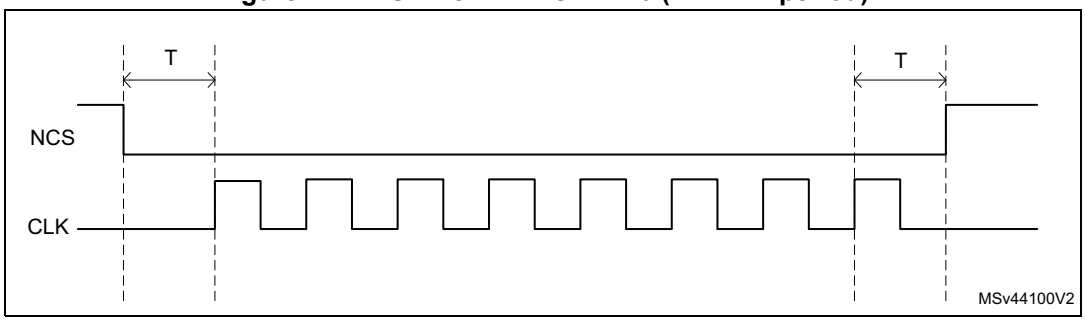
- After a memory-mapped read, the software must perform a synchronization barrier than an abort.

20.4.16 NCS behavior

By default, NCS is high, deselecting the external device. NCS falls before an operation begins and rises as soon as it finishes.

When CKMODE = 0 (clock mode 0: CLK stays low when no operation is in progress), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

Figure 77. NCS when CKMODE = 0 (T = CLK period)



Note: Clock mode 3 is not supported.

In dual-memory configuration, the NCS2 behaves exactly the same as the NCS1. The quad-SPI/octal-SPI memory 2 can then use the NCS1, and the NCS2 pin can be used for other functions.

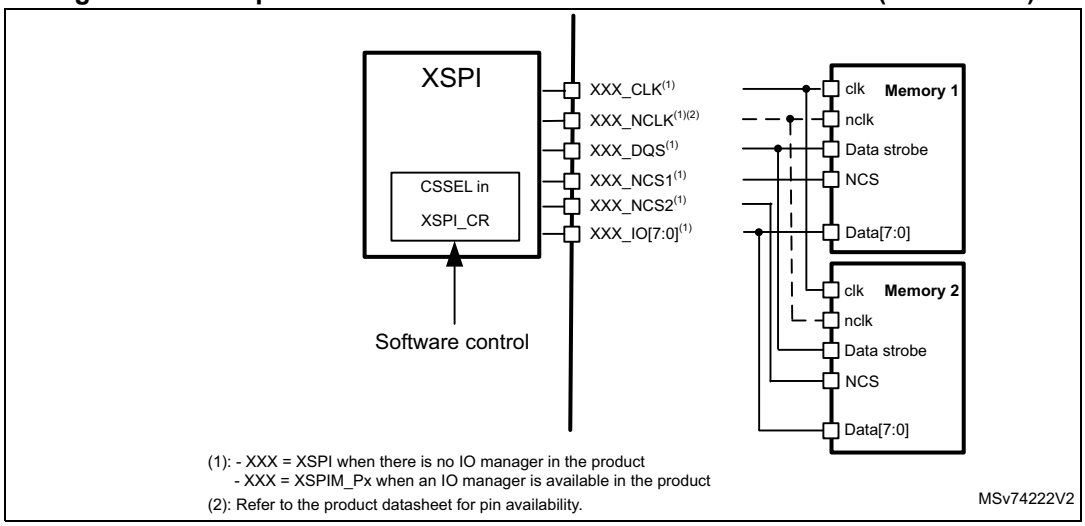
20.4.17 Software control of two external memories

One XSPI instance is able to control the generation of the two chip-select signals (NCS1 and NCS2). This allows two external memories to be connected to the same I/O port driven by the XSPI.

The XSPI generates one NCS signal and the user defines how the NCS is directed to NCS1 or NCS2, by setting the CSSEL bit in XSPI_CR. Intended operation is semistatic commutation, such as starting an application reading an external flash memory and then, for the rest of the application, use an external SRAM connected to the same I/O bus.

From a memory-map point of view, these two memories are located in the same address range.

Figure 78. Example of software control of two external memories (octal mode)



20.5 Address alignment and data number

The table below summarizes the effect of the address alignment and programmed data number depending on the use case.

Table 154. Address alignment cases

Memory type	Transaction type	Constraint on address(1)	Impact if constraint on address not respected	Constraint on number of bytes(1)	Impact if constraint on bytes not respected
Single, dual, quad flash or SRAM (DMM = 0)	IND(2) read	None	None	None	None
	MM(3) read				
	IND write				
Single, dual, quad flash or SRAM (DMM = 1)	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
Octal flash in SDR mode	IND read	None	None	None	None
	MM read				
	IND write				
Octal memory in DTR mode without WDM(6)	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
Octal flash or RAM in DTR mode with WDM	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write(7)				

1. To be respected by the software.
2. IND = indirect mode.

3. MM = memory-mapped mode.
4. Extra data at transfer start.
5. Extra data at transfer end.
6. WDM = write data mask.
7. If the FTHRES bitfield is set to the maximum value with DLR value greater than the data burst length, and if the DMA is enabled or the interrupt based on FIFO THRESHOLD Flag is enabled (FTF), the address must be modulo 2 aligned in DTR mode when the initiator (DMA, CPU, ...) is writing the data with a burst length equal to the FIFO size.

20.6 XSPI interrupts

An interrupt can be produced on the following events:

- Timeout
- FIFO threshold
- Transfer complete
- Transfer error

Separate interrupt enable bits are available to provide more flexibility.

Table 155. XSPI interrupt requests

Interrupt event	Event flag	Enable control bit
Timeout	TOF	TOIE
FIFO threshold	FTF	FTIE
Transfer complete	TCF	TCIE
Transfer error	TEF	TEIE

20.7 XSPI registers

20.7.1 XSPI control register (XSPI_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MSEL	FMODE[1:0]		Res.	Res.	NOPREF	CSSEL	Res.	Res.	BERRIE	TOIE	Res.	FTIE	TCIE	TEIE
	rw	rw	rw			rw	rw			rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	FTHRES[4:0]					Res.	DMM	Res.	ADOFFEN	TCEN	DMAEN	ABORT	EN
			rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **MSEL**: External memory select

This bit selects the external memory to be addressed in single-, dual-, quad-SPI mode in single-memory configuration (when DMM = 0).

0: External memory 1 selected (data exchanged over IO[3:0])

1: External memory 2 selected (data exchanged over IO[7:4])

This bit is ignored when DMM = 1 or when octal-SPI mode is selected.

Note: This bitfield can be modified only when BUSY = 0.

Bits 29:28 **FMODE[1:0]**: Functional mode

This bitfield defines the XSPI functional mode of operation.

00: Indirect-write mode

01: Indirect-read mode

10: No action derived from this setting

11: Memory-mapped mode

If DMAEN = 1 already, then the DMA controller for the corresponding channel must be disabled before changing the FMODE[1:0] value.

Note: This bitfield can be modified only when BUSY = 0.

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **NOPREF**: No prefetch data

This bit disables the automatic prefetch in the external memory.

0: Automatic prefetch enabled

1: Automatic prefetch disabled

Bit 24 **CSSEL**: Chip-select selection

This bit indicates if it must copy NCS to NCS1 or NCS2.

0: NCS1 active

1: NCS2 active

Note: This bit can be modified only when BUSY = 0.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **BERRIE**: Bus error interrupt enable

This bit enables the bus error interrupt (if Bus error bit feature is available in the product implementation table).

0: Interrupt disabled

1: Interrupt enabled

Bit 20 **TOIE**: Timeout interrupt enable

This bit enables the timeout interrupt.

0: Interrupt disabled

1: Interrupt enabled

Bit 19 Reserved, must be kept at reset value.

Bit 18 **FTIE**: FIFO threshold interrupt enable

This bit enables the FIFO threshold interrupt.

0: Interrupt disabled

1: Interrupt enabled

Bit 17 **TCIE**: Transfer complete interrupt enable

This bit enables the transfer complete interrupt.

0: Interrupt disabled

1: Interrupt enabled

Bit 16 **TEIE**: Transfer error interrupt enable

This bit enables the transfer error interrupt.

0: Interrupt disabled

1: Interrupt enabled

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **FTHRES[4:0]**: FIFO threshold level

This bitfield defines, in indirect mode, the threshold number of bytes in the FIFO that causes the FIFO threshold flag FTF in XSPI_SR, to be set.

00000: FTF is set if there are one or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are one or more valid bytes can be read from the FIFO in indirect-read mode.

00001: FTF is set if there are two or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are two or more valid bytes can be read from the FIFO in indirect-read mode.

...

11111: FTF is set if there are 32 free bytes available to be written to in the FIFO in indirect-write mode, or if there are 32 valid bytes can be read from the FIFO in indirect-read mode.

Note: If DMAEN = 1, the DMA controller for the corresponding channel must be disabled before changing the FTHRES[4:0] value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **DMM**: Dual-memory configuration

This bit activates the dual-memory configuration, where two external devices are used simultaneously to double the throughput and the capacity

0: Dual-memory configuration disabled

1: Dual-memory configuration enabled

Note: This bit can be modified only when BUSY = 0.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **ADOFFEN**: Address offset enable

An offset may be applied on the memory address. The offset value is set into ADOFF[4:0].

0: Address offset disabled

1: Address offset enabled

Note: This bit can be modified only when BUSY = 0. This bit is present only if the address offset feature is available in the product implementation table.

Bit 3 **TCEN**: Timeout counter enable

This bit is valid only when the memory-mapped mode (FMODE[1:0] = 11) is selected. This bit enables the timeout counter.

0: The timeout counter is disabled, and thus the chip-select (NCS) remains active indefinitely after an access in memory-mapped mode.

1: The timeout counter is enabled, and thus the chip-select is released in the memory-mapped mode after TIMEOUT[15:0] cycles of external device inactivity.

Note: This bit can be modified only when BUSY = 0.

Bit 2 **DMAEN**: DMA enable

In indirect mode, the DMA can be used to input or output data via XSPI_DR. DMA transfers are initiated when FTF is set.

0: DMA disabled for indirect mode

1: DMA enabled for indirect mode

Note: Resetting the DMAEN bit while a DMA transfer is ongoing, breaks the handshake with the DMA. Do not write this bit during DMA operation.

Bit 1 **ABORT**: Abort request

This bit aborts the ongoing command sequence. It is automatically reset once the abort is completed. This bit stops the current transfer.

0: No abort requested

1: Abort requested

Note: This bit is always read as 0.

Bit 0 **EN**: Enable

This bit enables the XSPI.

0: XSPI disabled

1: XSPI enabled

Note: The DMA request can be aborted without having received the ACK in case this EN bit is cleared during the operation.
In case this bit is set to 0 during a DMA transfer, the REQ signal to DMA returns to inactive state without waiting for the ACK signal from DMA to be active.

20.7.2 XSPI device configuration register 1 (XSPI_DCR1)

Address offset: 0x008

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADOFF[4:0]					MTYP[2:0]			Res.	Res.	Res.	DEVSIZE[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CSHT[5:0]						Res.	Res.	Res.	Res.	DLYBYP	Res.	FRCK	CKMODE
		rw	rw	rw	rw	rw	rw					rw		rw	rw

Bits 31:27 **ADOFF[4:0]**: Address offset

This bitfield defines the memory address offset value added to the current address requested to be sent by the controller to hit the external memory using the following formula:
Offset value = $2^{[ADOFF+8]}$. (modulo 256 bytes)
Maximum allowed value is 19 since the memory-mapped mode is limited to 256 Mbytes address space. For greater value, the address offset will not be applied to the address sent by the controller.
This bitfield is relevant only if ADOFFEN is set and if FMODE[1:0] = 11 (memory-mapped mode) and if the address offset feature is available in the product implementation table.

Bits 26:24 **MTYP[2:0]**: Memory type

This bitfield indicates the type of memory to be supported.

000: Micron mode, D0/D1 ordering in DTR 8-data-bit mode. regular-command protocol in single-, dual-, quad- and octal-SPI modes.

Note: In this mode, DQS signal polarity is inverted with respect to the memory clock signal. This is the default value and care must be taken to change MTYP[2:0] for memories different from Micron.

001: Macronix mode, D1/D0 ordering in DTR 8-data-bit mode. Regular-command protocol in single-, dual-, quad-, and octal-SPI modes.

010: Standard mode

011: Macronix RAM mode, D1/D0 ordering in DTR 8-data-bit mode. Regular-command protocol in single-, dual-, quad-, and octal-SPI modes with dedicated address mapping (address is built with row and column to fit with Macronix requirements).

Other: Reserved

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **DEVSIZE[4:0]**: Device size

This bitfield defines the size of the external device using the following formula:

Number of bytes in device = $2^{[DEVSIZE+1]}$.

DEVSIZE + 1 is effectively the number of address bits required to address the external device. The device capacity can be up to 4 Gbytes (addressed using 32-bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

In regular-command protocol, if DMM = 1, DEVSIZE[4:0] must reflect the total capacity of the two devices together considering the above formula (DEVSIZE[4:0] value is so equal to one of the two memory capacities).

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **CSHT[5:0]**: Chip-select high time

CSHT + 1 defines the minimum number of CLK cycles where the chip-select (NCS) must remain high between commands issued to the external device.

0x0: NCS stays high for at least 1 cycle between external device commands.

0x1: NCS stays high for at least 2 cycles between external device commands.

...

0x3F: NCS stays high for at least 64 cycles between external device commands.

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **DLYBYP**: Delay block bypass

0: The internal sampling clock (called feedback clock) or the DQS data strobe external signal is delayed by the delay block (for more details on this block, refer to the dedicated section of the reference manual as it is not part of the peripheral).

1: The delay block is bypassed, so the internal sampling clock or the DQS data strobe external signal is not affected by the delay block. The delay is shorter than when the delay block is not bypassed, even with the delay value set to minimum value in delay block.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **FRCK**: Free running clock

This bit configures the free running clock.

0: CLK is not free running.

1: CLK is free running (always provided).

Note: Free running clock mode is intended for delay calibration only. No memory or other device access is possible when FRCK is set. Once the bit has been set to 1, the BSY bit is set, the NCS signal remains set to 1 and the clock is sent. The only way to stop the clock is to perform an abort (which clears the BSY bit). Then, the FRCK bit must be cleared by software to allow controller transactions to take place with the external memory.

Bit 0 **CKMODE**: Clock mode 0

This bit is read-only and is read as zero.

20.7.3 XSPI device configuration register 2 (XSPI_DCR2)

Address offset: 0x00C

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRESCALER[7:0]**: Clock prescaler

This bitfield defines the scaler factor for generating the CLK based on the kernel clock (value + 1).

0: $F_{CLK} = F_{KERNEL}$, kernel clock used directly as XSPI CLK (prescaler bypassed). In this case, if the DTR mode is used, it is mandatory to provide to the XSPI a kernel clock that has 50% duty-cycle.

1: $F_{CLK} = F_{KERNEL}/2$

2: $F_{CLK} = F_{KERNEL}/3$

...

255: $F_{CLK} = F_{KERNEL}/256$

For odd clock division factors, the CLK duty cycle is not 50 %. The clock signal remains low one cycle longer than it stays high.

20.7.4 XSPI status register (XSPI_SR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FLEVEL[5:0]						Res.	BERRF	BUSY	TOF	Res.	FTF	TCF	TEF
		r	r	r	r	r	r		r	r	r		r	r	r

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **FLEVEL[5:0]**: FIFO level

This bitfield gives the number of valid bytes that are being held in the FIFO. FLEVEL = 0 when the FIFO is empty, and 32 when it is full.

In automatic status-polling mode, FLEVEL is zero.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **BERRF**: Bus error flag

This bit is set when a bus error occurs. It is cleared by writing 1 to CBERRF. This bit has a meaning if Bus error bit feature is available in the product implementation table.

Bit 5 **BUSY**: Busy

This bit is set when an operation is ongoing. It is cleared automatically when the operation with the external device is finished and the FIFO is empty.

Bit 4 **TOF**: Timeout flag

This bit is set when timeout occurs. It is cleared by writing 1 to CTOF.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FTF**: FIFO threshold flag

In indirect mode, this bit is set when the FIFO threshold has been reached, or if there is any data left in the FIFO after the reads from the external device are complete.

It is cleared automatically as soon as the threshold condition is no longer true.

Bit 1 **TCF**: Transfer complete flag

This bit is set in indirect mode when the programmed number of data has been transferred or in any mode when the transfer has been aborted.It is cleared by writing 1 to CTCF.

Bit 0 **TEF**: Transfer error flag

This bit is set in indirect mode when an invalid address is being accessed in indirect mode.

It is cleared by writing 1 to CTEF.

20.7.5 XSPI flag clear register (XSPI_FCR)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBERRF	Res.	CTOF	Res.	Res.	CTCF	CTEF
									w		w			w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CBERRF**: Clear bus error flag
Writing 1 clears the BERRF flag in the XSPI_SR register (if Bus error bit feature is available in the product implementation table).

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CTOF**: Clear timeout flag
Writing 1 clears the TOF flag in the XSPI_SR register.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **CTCF**: Clear transfer complete flag
Writing 1 clears the TCF flag in the XSPI_SR register.

Bit 0 **CTEF**: Clear transfer error flag
Writing 1 clears the TEF flag in the XSPI_SR register.

20.7.6 XSPI data length register (XSPI_DLR)

Address offset: 0x040

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **DL[31:0]**: Data length

Number of data to be retrieved (value+1) in indirect mode.
All 1's in indirect mode means undefined length, where XSPI continues until the end of the memory, as defined by DEVSIZE.
0x0000_0000: 1 byte is to be transferred.
0x0000_0001: 2 bytes are to be transferred.
0x0000_0002: 3 bytes are to be transferred.
0x0000_0003: 4 bytes are to be transferred.
...
0xFFFF_FFFD: 4,294,967,294 (4G-2) bytes are to be transferred.
0xFFFF_FFFE: 4,294,967,295 (4G-1) bytes are to be transferred.
0xFFFF_FFFF: undefined length; all bytes, until the end of the external device, (as defined by DEVSIZE) are to be transferred. Continue reading indefinitely if DEVSIZE = 0x1F.
DL[0] is stuck at 1 in dual-memory configuration (DMM = 1) even when 0 is written to this bit, thus assuring that each access transfers an even number of bytes.
This bitfield has no effect in memory-mapped mode.

20.7.7 XSPI address register (XSPI_AR)

Address offset: 0x048

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0 and FMODE ≠ 11.

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

Bits 31:0 **ADDRESS[31:0]**: Address

Address to be sent to the external device. In dual-memory configuration, AR[0] is forced to 0.

Caution: Some memory specifications consider that each address corresponds to a 16-bit value. XSPI considers that each address corresponds to an 8-bit value. So the software needs to multiply the address by two when accessing the memory registers.

20.7.8 XSPI data register (XSPI_DR)

Address offset: 0x050

Reset value: 0x0000 0000

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

Bits 31:0 **DATA[31:0]**: Data

Data to be sent/received to/from the external SPI device

In indirect-write mode, data written to this register is stored on the FIFO before it is sent to the external device during the data phase. If the FIFO is too full, a write operation is stalled until the FIFO has enough space to accept the amount of data being written.

In indirect-read mode, reading this register gives (via the FIFO) the data that was received from the external device. If the FIFO does not have as many bytes as requested by the read operation and if BUSY = 1, the read operation is stalled until enough data is present or until the transfer is complete, whichever happens first.

Word, half-word, and byte accesses to this register are supported. In indirect-write mode, a byte write adds 1 byte to the FIFO, a half-word write 2 bytes, and a word write 4 bytes. Similarly, in indirect-read mode, a byte read removes 1 byte from the FIFO, a halfword read 2 bytes, and a word read 4 bytes. Accesses in indirect mode must be aligned to the bottom of this register: A byte read must read DATA[7:0] and a half-word read must read DATA[15:0].

20.7.9 XSPI communication configuration register (XSPI_CCR)

Address offset: 0x100

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable

This bit enables the data strobe management.

0: DQS disabled

1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate

This bit sets the DTR mode for the data phase.

0: DTR mode disabled for data phase

1: DTR mode enabled for data phase

Bits 26:24 **DMODE[2:0]**: Data mode

This bitfield defines the data phase mode of operation.

000: No data

001: Data on a single line

010: Data on two lines

011: Data on four lines

100: Data on eight lines

Other: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate- byte double transfer rate
This bit sets the DTR mode for the alternate-byte phase.
0: DTR mode disabled for the alternate-byte phase
1: DTR mode enabled for the alternate-byte phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
Other: Reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for the address phase
1: DTR mode enabled for the address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Other: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for the instruction phase
1: DTR mode enabled for the instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Other: Reserved

20.7.10 XSPI timing configuration register (XSPI_TCR)

Address offset: 0x108

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
											rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SSHIFT**: Sample shift
By default, the XSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.
This bit allows the data to be sampled later in order to consider the external signal delays.
0: No shift
1: 1/2 cycle shift
The software must ensure that SSHIFT = 0 when the data phase is configured in DTR mode (when DDTR = 1.)

Bit 29 Reserved, must be kept at reset value.

Bit 28 **DHQC**: Delay hold quarter cycle
0: No delay hold
1: 1/4 cycle hold

Bits 27:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles
This bitfield defines the duration of the dummy phase according to the memory latency.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

20.7.11 XSPI instruction register (XSPI_IR)

Address offset: 0x110

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

20.7.12 XSPI alternate bytes register (XSPI_ABR)

Address offset: 0x120

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes
Optional data to be sent to the external SPI device right after the address.

20.7.13 XSPI low-power timeout register (XSPI_LPTR)

Address offset: 0x130

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TIMEOUT[15:0]**: Timeout period

After each access in memory-mapped mode, the XSPI prefetches the subsequent bytes and hold them in the FIFO.

This bitfield indicates how many CLK cycles the XSPI waits after the clock becomes inactive and until it raises the NCS, putting the external device in a lower-consumption state.

20.7.14 XSPI register map

Table 156. XSPI register map and reset values

Offset	Register name	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
0x000	XSPI_CR	Res.	MSEL	FMDOE[1:0]		Res.	Res.	NOPREF	CSSEL	Res.	Res.	BERRIE	TOIE	Res.	FTIE	TCIE	TEIE	Res.	Res.	Res.	FTHRES[4:0]					Res.	DMM	Res.	ADOFFEN	TCEN	DMAEN	ABORT	EN
	Reset value		0	0	0			0	0			0	0		0	0	0				0	0	0	0	0		0		0	0	0	0	0
0x004	Reserved																																
0x008	XSPI_DCR1	ADOFF[4:0]					MTYP [2:0]			Res.	Res.	Res.	DEVSIZE[4:0]					Res.	Res.	CSHT[5:0]						Res.	Res.	Res.	Res.	DLYBYP	Res.	FRCK	CKMODE
	Reset value	0	0	0	0	0	0	0	0				0	0	0	0	0			0	0	0	0	0	0					0		0	0
0x00C	XSPI_DCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x010	XSPI_DCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSBOUND[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value												0	0	0	0	0																
0x020	XSPI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLEVEL[5:0]						Res.	BERRF	BUSY	TOF	Res.	FTF	TCF	TEF
	Reset value																			0	0	0	0	0	0		0	0	0		0	0	0
0x024	XSPI_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBERRF	Res.	CTOF	Res.	Res.	CTCF	CTEF
	Reset value																										0		0			0	0
0x028-0x03C	Reserved																																
0x040	XSPI_DLR	DL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	Reserved	Reserved																															
0x048	XSPI_AR	ADDRESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	Reserved	Reserved																															
0x050	XSPI_DR	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054-0x0FC	Reserved																																

Table 156. XSPI register map and reset values (continued)

Offset	Register name	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
0x100	XSPI_CCR	Res.	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE [2:0]		
	Reset value			0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x104	Reserved																																
0x108	XSPI_TCR	Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value		0		0																								0	0	0	0	0
0x10C	Reserved																																
0x110	XSPI_IR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x114-0x11C	Reserved																																
0x120	XSPI_ABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x124-0x12C	Reserved																																
0x130	XSPI_LPTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMEOUT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

21 Delay block (DLYB)

21.1 DLYB introduction

The delay block (DLYB) is used to generate an output clock that is dephased from the input clock. The phase of the output clock must be programmed by the user application. The output clock is then used to clock the data received by another peripheral such as an XSPI interface.

The delay is voltage- and temperature-dependent, that may require the application to re-configure and recenter the output clock phase with the receive data.

21.2 DLYB main features

The delay block has the following features:

- Input clock frequency ranging from 25 MHz to the maximum frequency supported by the communication interface (see datasheet)
- Up to 12 oversampling phases.

21.3 DLYB implementation

Table 157. STM32WBA2 features

DLYB associated peripheral	DLYBXS1
XSPI1	X

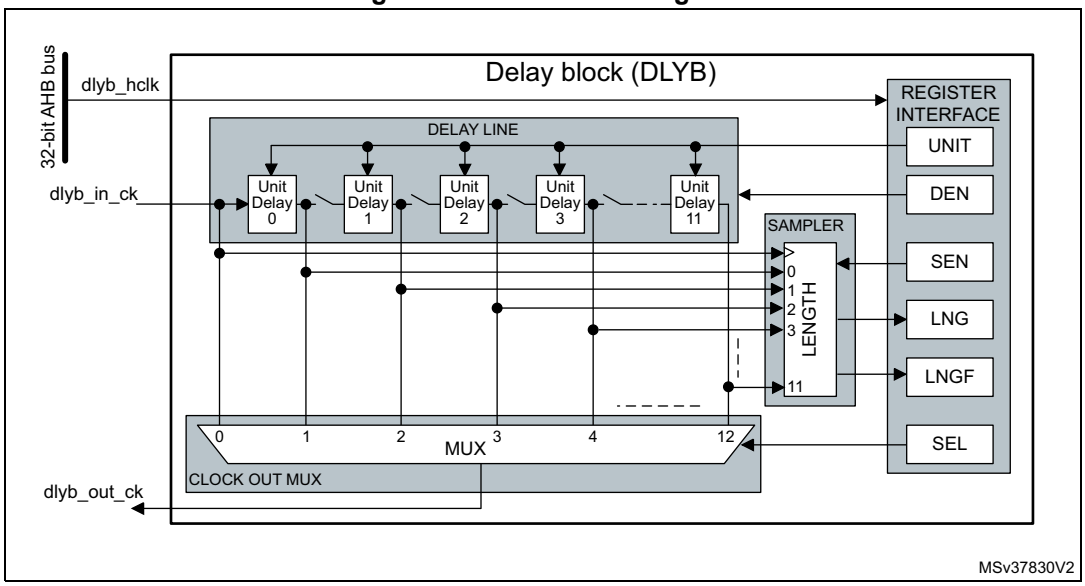
21.4 DLYB functional description

21.4.1 DLYB diagram

The delay block includes the following sub-blocks (shown in the figure below):

- register interface block providing AHB access to the DLYB registers
- delay line supporting the unit delays
- delay line length sampling
- output clock selection multiplexer

Figure 79. DLYB block diagram



21.4.2 DLYB pins and internal signals

Table 158 lists the DLYB internal signals.

Table 158. DLYB internal input/output signals

Signal name	Signal type	Description
dlyb_hclk	Digital input	Delay block register interface clock
dlyb_in_ck	Digital input	Delay block input clock
dlyb_out_ck	Digital output	Delay block output clock

Table 159. DLYB interconnection

Signal name	Source/destination
	XSPI1
dlyb_in_ck	DQS
dlyb_out_ck	DQS delayed for data sampling

21.4.3 General description

The delay block is enabled by setting the DEN bit in the DLYB control register (DLYB_CR). The length sampler is enabled through the SEN bit in DLYB_CR register.

When the delay block is enabled, the delay added by a unit delay is defined by the UNIT[6:0] field in the DLYB configuration register (DLYB_CFGR).

Note: UNIT[6:0] can be programmed only when the output clock is disabled (SEN = 1).

When the delay block is enabled, the output clock phase is selected through the SEL[3:0] field in DLYB_CFGR register.

Note: SEL can be programmed only when the output clock is disabled (SEN = 1).

The output clock can be de-phased over one input clock period by configuring the delay line length to span one period. The delay line length can be configured by enabling the length sampler through the SEN bit, that gives access to the delay line length (LNG[11:0]) and length valid flag (LNGF) in DLYB_CFGR.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced. This allows a smaller unit delay providing higher resolution.

Once the delay line length is configured, a dephased output clock can be selected by the output clock multiplexer. This is done through SEL[3:0]. The output clock is only available on the selected phase when SEN is set to 0.

The table below gives a summary of the delay block control.

Table 160. Delay block control

DEN	SEN	UNIT	SEL	LNG	LNGF	Output clock
0	0	Don't care	Don't care	Don't care	Don't care	Enabled (= Input clock)
x	1	Unit delay	Output clock phase	Length	Length flag	Disabled
1	0	Unit delay(1)	Output clock phase(2)	Don't care	Don't care	Enabled (= selected phase)

1. The unit delay can only be changed when SEN = 1.
2. The output clock phase can only be changed when SEN = 1.

21.4.4 Delay line length configuration procedure

LNG[11:0] is used to determine the delay line length with respect to the input clock period. The length must be configured so that one full input clock period is covered by the delay line length.

Note that despite the delay line has 12 unit delay elements, the following procedure description returns a length between 0 and 10, as the upper delay output value is used to ensure that the delay is calibrated over one full input clock cycle. Depending on the clock frequency and UNIT value, unit delay element 10 may also be truncated from the clock cycle length.

A clock input (free running clock) must be present during the whole tuning procedure.

To configure the delay line length to one period of the Input clock, follow the sequence below:

1. Enable the delay block by setting DEN bit to 1.
2. Enable the length sampling by setting SEN bit to 1.
3. Enable all delay cells by setting SEL[3:0] to 12.
4. For UNIT[6:0] = 0 to 127 (this step must be repeated until the delay line length is configured):
   a) Update the UNIT[6:0] value and wait till the length flag LNGF is set to 1.
   b) Read LNG[11:0].

   If (LNG[10:0] > 0) and (LNG[11] or LNG[10] = 0), the delay line length is configured to one input clock period.
5. Determine how many unit delays (N) span one input clock period: for N = 0 to 10, if LNG[N] = 1, the number of unit delays spanning the input clock period = N.
6. Disable the length sampling by clearing SEN to 0.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced smaller than one input clock period. This allows a smaller unit delay, providing a higher resolution spanning a shorter time interval.

21.4.5 Output clock phase configuration procedure

When the delay line length is configured to one input clock period, the output clock phase can be selected between the unit delays spanning one Input clock period.

Follow the steps below to select the output clock phase:

1. Disable the output clock and enable the access to the phase selection SEL[3:0] bits by setting SEN bit to 1.
2. Program SEL[3:0] with the desired output clock phase value.
3. Enable the output clock on the selected phase by clearing SEN to 0.

XSPI use case:

The delay block is used in conjunction with XSPI interface to allow shifting the input data sampling signal. This sampling signal can be the feedback clock or the data strobe (DQS) signal, which is delivered by certain type of devices. Note that in case DQS is used, the calibration procedure must be performed beforehand with a free running clock, as DQS is a discontinuous signal.

In case of SDR (single data rate) mode the user must typically shift the sampling signal by half period, so that the sampling edges are positioned in the middle of the valid data phase.

In case of DDR (dual data rate) mode, for which data are transitioning at start and middle of period, typical value must be close to N/4, once the calibration is completed.

In case of high frequencies and tight timing constraints, the delay setting granularity (10) might be too coarse. Since in most cases it is not necessary to have a possible delay value covering the whole sampling clock period, the "Unit" value can be overridden by application in order to improve the accuracy of the sampling edge position (example: providing a twice as small “Unit” gives twice better timing accuracy. The counterpart being that the maximum possible delay is divided by 2).

21.5 DLYB registers

All registers can be accessed in word, half-word and byte access.

21.5.1 DLYB control register (DLYB_CR)

Address offset: 0x000

Reset value: 0x0000 0000

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SEN:** Sampler length enable bit

0: Sampler length and register access to UNIT[6:0] and SEL[3:0] disabled, output clock enabled.

1: Sampler length and register access to UNIT[6:0] and SEL[3:0] enabled, output clock disabled.

Bit 0 **DEN:** Delay block enable bit

0: DLYB disabled.

1: DLYB enabled.

21.5.2 DLYB configuration register (DLYB_CFGR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LNGF	Res.	Res.	Res.	LNG[11:0]											
r				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	UNIT[6:0]							Res.	Res.	Res.	Res.	SEL[3:0]			
	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bit 31 **LNGF:** Length valid flag

This flag indicates when the delay line length value contained in LNG[11:0] is valid after UNIT[6:0] bits changed.

0: Length value in LNG is not valid.

1: Length value in LNG is valid.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **LNG[11:0]:** Delay line length value

These bits reflect the 12 unit delay values sampled at the rising edge of the input clock. The value is only valid when LNGF = 1.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **UNIT[6:0]:** Delay of a unit delay cell.
These bits can only be written when SEN = 1.
Unit delay = initial delay + UNIT[6:0] x delay step

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL[3:0]:** Phase for the output clock.
These bits can only be written when SEN = 1.
Output clock phase = input clock + SEL[3:0] x unit delay

21.5.3 DLYB register map

Table 161. DLYB register map and reset values

Offset	Register name	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
0x000	DLYB_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEN	DEN
	Reset value																															0	0
0x004	DLYB_CFGR	LNGF	Res.	Res.	Res.	LNG[11:0]												Res.	UNIT[6:0]							Res.	Res.	Res.	Res.	SEL[3:0]			
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0					0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

22 Analog-to-digital converter (ADC4)

22.1 ADC introduction

The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 10 multiplexed channels enabling it to measure signals from 7 external sources and 3 internal sources. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned 16-bit data register.

The analog watchdog feature enables the application to detect if the input voltage goes outside the user-defined higher or lower thresholds.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

A built-in hardware oversampler allows improving analog performances while off-loading the related computational burden from the CPU.

22.2 ADC main features

- High performance
  - 12-bit, 10-bit, 8-bit or 6-bit configurable resolution
  - ADC conversion time: 0.4 µs for 12-bit resolution (2.5 Msps), faster conversion times can be obtained by lowering resolution.
  - Self-calibration
  - Programmable sampling time
  - Data alignment with built-in data coherency
  - DMA support
- Low-power
  - The application can reduce the bus clock frequency for low-power operation while still keeping optimum ADC performance. For example, 0.4 µs conversion time is kept, whatever the bus clock frequency
  - Wait mode: prevents ADC overrun in applications with low bus clock frequency
  - Auto-off mode: ADC is automatically powered off except during the active conversion phase. This dramatically reduces the power consumption of the ADC.
- Autonomous mode
  - Conversion and DMA transfers supported in Stop mode
  - Wake-up from Stop on ADC interrupts
  - Enter and exit from Deep-power-down mode managed automatically
- Analog input channels
  - 7 external analog inputs
  - 1 channel for the internal temperature sensor (V_{SENSE})
  - 1 channel for the internal reference voltage (V_{REFINT})
  - 1 channel for the internal digital core voltage (V_{CORE})

- Start-of-conversion can be initiated:
  - By software
  - By hardware triggers with configurable polarity (timer events or GPIO input events)
- 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
- Interrupt generation at the end of sampling, end of conversion, end of sequence conversion, and in case of analog watchdog or overrun events
- Analog watchdog
- Oversampler
  - 16-bit data register
  - Oversampling ratio adjustable from 2 to 256x
  - Programmable data shift up to 8 bits
- ADC input range: $V_{SSA} \leq V_{IN} \leq V_{DDA}$

22.3 ADC implementation

Table 162. ADC features(1)

ADC modes/features	ADC4
Resolution	12 bits
Maximum sampling speed for 12-bit resolution	2.5 Msps
Hardware offset calibration	X
Hardware linearity calibration	-
Single-ended inputs	X
Differential inputs	-
Injected channel conversion	-
Oversampling	up to x256
Data register	16 bits
DMA support	X
Parallel data output to MDF	-
Dual mode	-
Autonomous mode	X
Offset compensation	-
Gain compensation	-
Number of analog watchdogs	3
Wake up from Stop mode	X

1. Note: ‘X’ = supported, ‘-’= not supported.

Table 163. Memory location of the temperature sensor calibration values

Name	Description	Memory address
TS_CAL1	Temperature sensor 12-bit raw data acquired by ADC4 at 30 °C (± 5 °C), $V_{DDA} = V_{REF+}$ = 3.0 V (±10 mV)	0x0BF8 D710 - 0xBF8 D711
TS_CAL2	Temperature sensor 12-bit raw data acquired by ADC4 at 130 °C (± 5 °C), $V_{DDA} = V_{REF+}$ = 3.0 V (±10 mV)	0x0BF8 D742 - 0xBF8 D743

Table 164. Memory location of the internal reference voltage sensor calibration value

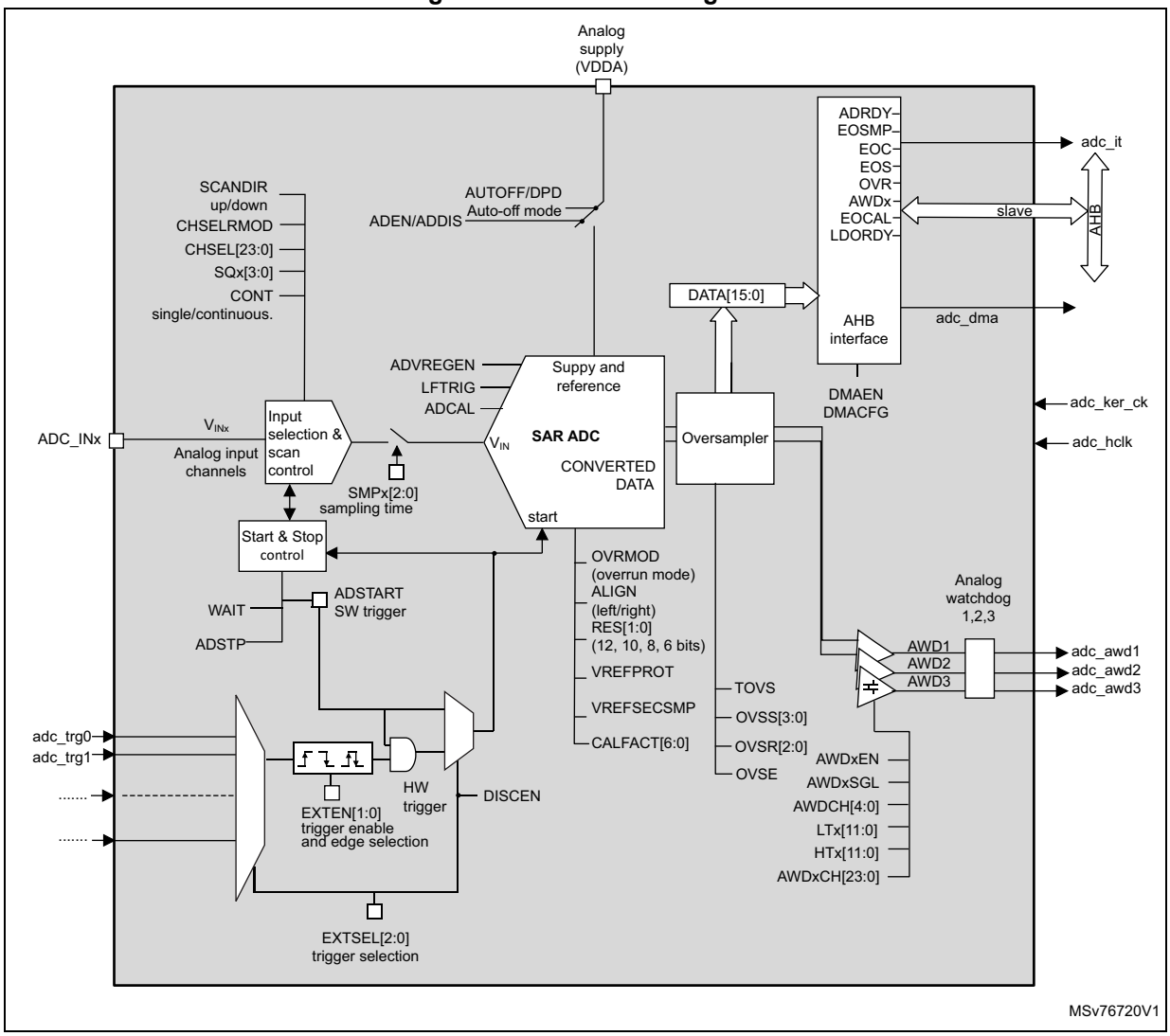
Name	Description	Memory address
VREFINT_CAL	12-bit raw data acquired by ADC4 at 30 °C (± 5 °C), $V_{DDA} = V_{REF+}$ = 3.0 V (±10 mV)	0x0BF8 D7A5 - 0xBF8 D7A6

22.4 ADC functional description

22.4.1 ADC block diagram

Figure 80 shows the ADC block diagram and *Table 165* gives the ADC pin description.

Figure 80. ADC block diagram



22.4.2 ADC pins and internal signals

Table 165. ADC input/output pins

Pin name	Signal type	Description
VDDA	Input, analog power supply	Analog power supply and positive reference voltage for the ADC, $V_{DDA} \geq V_{DD}$

Table 165. ADC input/output pins (continued)

Pin name	Signal type	Description
VSSA	Input, analog supply ground	Ground for analog power supply, equal to V_{SS}.
ADC_INx	Analog input signals	7 external analog input channels.

Table 166. ADC internal input/output signals

Internal signal name	Signal type	Description
V_{IN}[x]	Analog inputs	Analog input channels connected either to internal channels or to ADC_INx external channels.
adc_trgx	Inputs	ADC conversion triggers.
adc_awdx	Output	Internal analog watchdog output signal connected to on-chip timers (x = Analog watchdog number = 1,2,3).
adc_it	Output	ADC interrupt.
adc_hclk	Input	AHB clock.
adc_ker_ck	Input	ADC kernel clock input from the RCC block.
adc_dma	Output	ADC DMA request

Table 167. ADC interconnection

Signal name	Source/destination
ADC4 V_{IN}[13]	V_{SENSE} (internal temperature sensor output voltage)
ADC4 V_{IN}[0]	V_{REFINT} (buffered voltage from internal reference voltage)
ADC4 V_{IN}[12]	V_{CORE} (internal logic supply voltage).
adc_trg0	Reserved
adc_trg1	Reserved
adc_trg2	tim2_trgo
adc_trg3	Reserved
adc_trg4	Reserved
adc_trg5	lptim1_ch1
adc_trg6	Reserved
adc_trg7	exti15

22.4.3 ADC voltage regulator (ADVREGEN)

The ADC has a specific internal voltage regulator which must be enabled and stable before using the ADC.

The ADC internal voltage regulator can be enabled by setting ADVREGEN bit to 1 in the ADC_CR register. The software must wait for the ADC voltage regulator startup time ($t_{ADCVREG_SETUP}$) before launching a calibration or enabling the ADC. The LDO status can be verified by checking the LDORDY bit in ADC_ISR register.

After ADC operations are complete, the ADC can be disabled (ADEN = 0). It is then possible to save additional power by disabling the ADC voltage regulator (refer to *Section : ADC voltage regulator disable sequence*).

Note: When the internal voltage regulator is disabled, the internal analog calibration factor is reset, and a new calibration must be performed.

ADC voltage regulator enable sequence

To enable the ADC voltage regulator, follow the sequence below:

1. Clear the LDORDY bit in ADC_ISR register by programming this bit to 1.
2. Set the ADVREGEN bit to 1 in ADC_CR register.
3. Wait until LDORDY = 1 in the ADC_ISR register (LDORDY is set after the ADC voltage regulator startup time). This can be handled by interrupt if the interrupt is enabled by setting the LDORDYIE bit in the ADC_IER register.

ADC voltage regulator disable sequence

To disable the ADC voltage regulator, follow the sequence below:

1. Make sure that the ADC is disabled (ADEN = 0).
2. Clear ADVREGEN bit in ADC_CR register.
3. Clear the LDORDY bit in ADC_ISR register by programming this bit to 1(optional),

22.4.4 Calibration (ADCAL)

The ADC has a calibration feature. During the procedure, the ADC calculates a calibration factor which is internally applied to the ADC until the next ADC power-off. The application must not use the ADC during calibration and must wait until it is complete.

The calibration must be performed before starting analog-to-digital conversion. It removes the offset error which may vary from chip to chip due to process variation, supply voltage and temperature.

The calibration is initiated by software by setting bit ADCAL to 1. It can be initiated only when all the following conditions are met:

- the ADC voltage regulator is enabled (ADVREGEN = 1 and LDORDY = 1),
- the ADC is disabled (ADEN = 0), and
- the auto-off mode is disabled (AUTOFF = 0).

ADCAL bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon the calibration completes. After this, the calibration factor can be read from the ADC_DR register (from bits 6 to 0).

The internal analog calibration is kept if the ADC is disabled (ADEN = 0). When the ADC operating conditions change (V_{DDA} changes are the main contributor to ADC offset variations and temperature change to a lesser extend), it is recommended to re-run a calibration cycle. It is recommended to recalibrate when V_{DDA} voltage changed more than 10%.

The calibration factor is lost in the following cases:

- The power supply is removed from the ADC (for example when the product enters Standby mode).
- The ADC peripheral is reset.

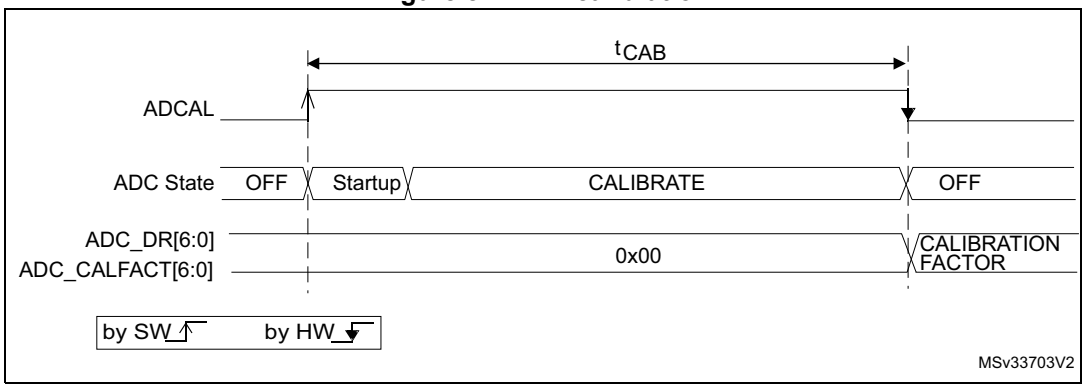
The calibration factor is lost each time power is removed from the ADC (for example when the product enters Standby mode). Still, it is possible to save and restore the calibration factor by software to save time when re-starting the ADC (as long as temperature and voltage are stable during the ADC power-down).

The calibration factor can be written if the ADC is enabled but not converting (ADEN = 1 and ADSTART = 0). Then, at the next start of conversion, the calibration factor is automatically injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion.

Software calibration procedure

1. Ensure that ADEN = 0, ADVREGEN = 1, AUTOFF = 0, DPD = 0, and DMAEN = 0.
2. Set ADCAL = 1.
3. Wait until ADCAL = 0 (or until EOCAL = 1). This can be handled by interrupt if the interrupt is enabled by setting the EOCALIE bit in the ADC_IER register
4. The calibration factor can be read from bits 6:0 of ADC_DR or ADC_CALFACT registers.

Figure 81. ADC calibration

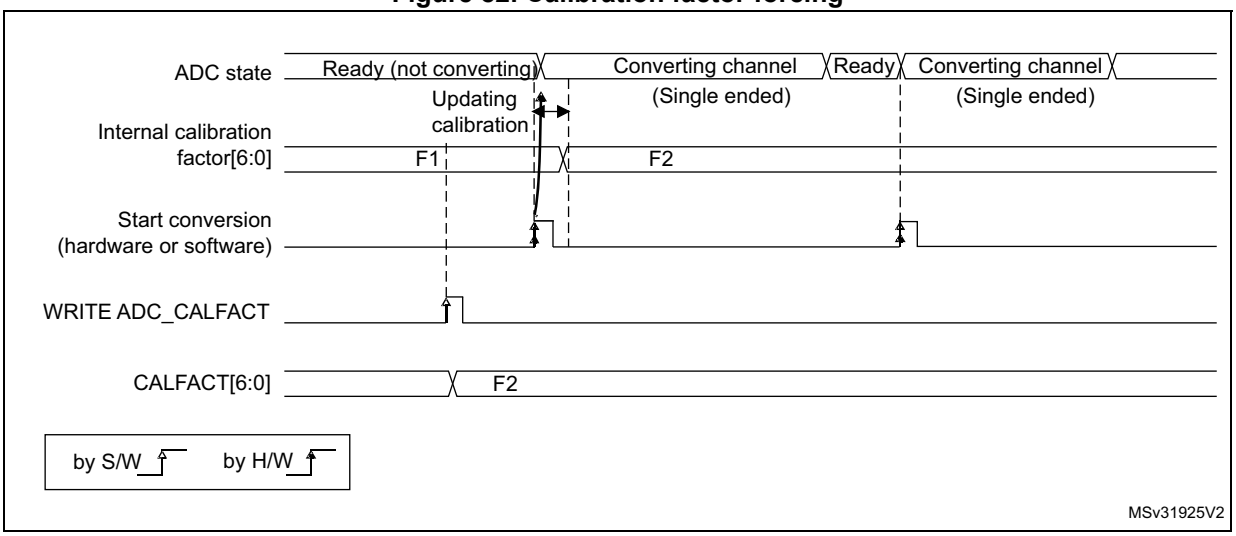


1. Refer to the device datasheet for the value of t_{CAB}.

Calibration factor forcing software procedure

1. Ensure that ADEN = 1 and ADSTART = 0 (ADC started with no conversion ongoing).
2. Write ADC_CALFACT with the saved calibration factor.
3. The calibration factor is used as soon as a new conversion is launched.

Figure 82. Calibration factor forcing



22.4.5 ADC on-off control (ADEN, ADDIS, ADRDY)

At power-up, the ADC is disabled and put in power-down mode (ADEN = 0).

As shown in *Figure 83*, the ADC needs a stabilization time of t_{STAB} 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. 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 to 1 (refer to *Section 22.4.16: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN)*) or when an external trigger event occurs if triggers are enabled.

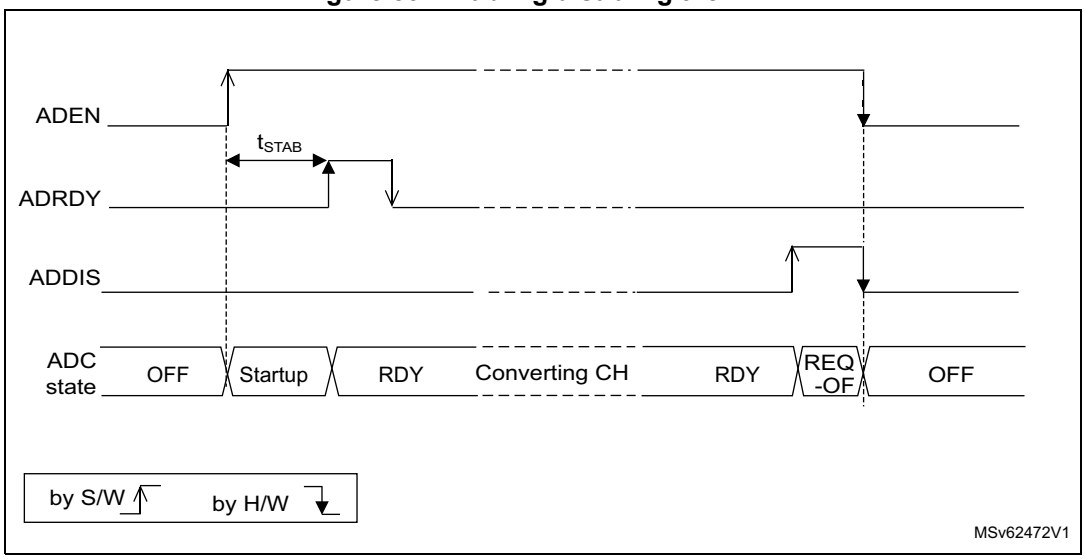
Follow the procedure below to enable the ADC:

1. Clear the ADRDY bit in ADC_ISR register by programming this bit to 1.
2. Set ADEN = 1 in the ADC_CR register.
3. Wait until ADRDY = 1 in the ADC_ISR 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_IER register.

Follow the procedure below to disable the ADC:

1. Check that ADSTART = 0 in the ADC_CR register to ensure that no conversion is ongoing. If the software trigger mode was used, stop the software trigger mode by writing 1 to the ADSTP bit of the ADC_CR register and waiting until this bit is read at 0.
2. Set ADDIS = 1 in the ADC_CR register.
3. If required by the application, wait until ADEN = 0 in the ADC_CR register, indicating that the ADC is fully disabled (ADDIS is automatically reset once ADEN = 0).
4. Clear the ADRDY bit in ADC_ISR register by programming this bit to 1 (optional).

Figure 83. Enabling/disabling the ADC



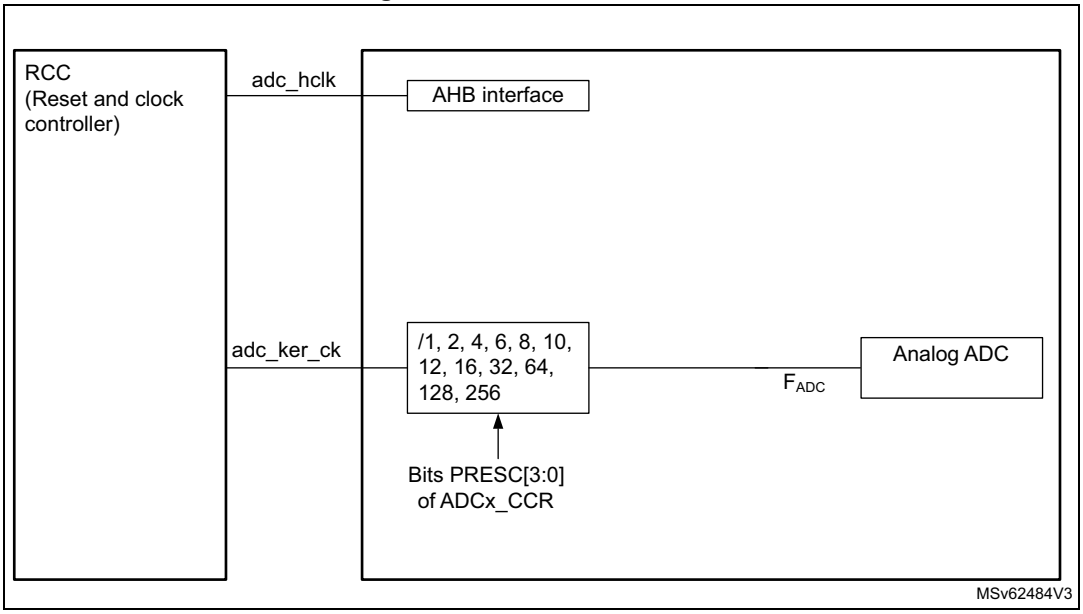
Note: In auto-off mode (AUTOFF = 1) the power-on/off phases are performed automatically, by hardware and the ADRDY flag is not set.

Caution: The ADEN bit cannot be set while the ADCAL bit is set, and during four ADC clock cycles after the ADCAL bit is cleared by the hardware (end of calibration).

22.4.6 ADC clock (PRESC[3:0])

The ADC has a dual clock-domain architecture, so that the ADC can be fed with a clock (ADC asynchronous clock) independent from the bus clock.

Figure 84. ADC clock scheme



1. Refer to *Section Reset and clock control (RCC)* for how the bus clock and ADC asynchronous clock are enabled.

The adc_ker_ck input clock can be selected between different clock sources (see *Figure 84: ADC clock scheme*). This selection is done in the RCC (refer to th RCC section for more information):

- The ADC clock can be provided by an internal or external clock source, which is independent and asynchronous with the bus clock.
- The ADC clock can be derived from the bus clock by selecting the adc_ker_ck as bus clock.

Option a) has the advantage of reaching the maximum ADC clock frequency whatever the clock scheme selected. The ADC clock can eventually be divided by a programmable ratio of 1, 2, 4, 6, 8, 10, 12, 16, 32, 64, 128 or 256, configured through PRESC[3:0] bits in the ADCx_CCR register.

Option b) has the advantage of bypassing the clock domain resynchronizations. This can be useful when the ADC is triggered by a timer and if the application requires that the ADC is precisely triggered without any uncertainty (otherwise, an uncertainty of the trigger instant is added by the resynchronizations between the two clock domains).

Table 168. Latency between trigger and start of conversion(1)

ADC clock source	Latency between the trigger event and the start of conversion
Clock different from bus clock	Latency is not deterministic (jitter)
Bus clock divided by 2	Latency is deterministic (no jitter) and equal to 4 ADC clock cycles

Table 168. Latency between trigger and start of conversion(1)

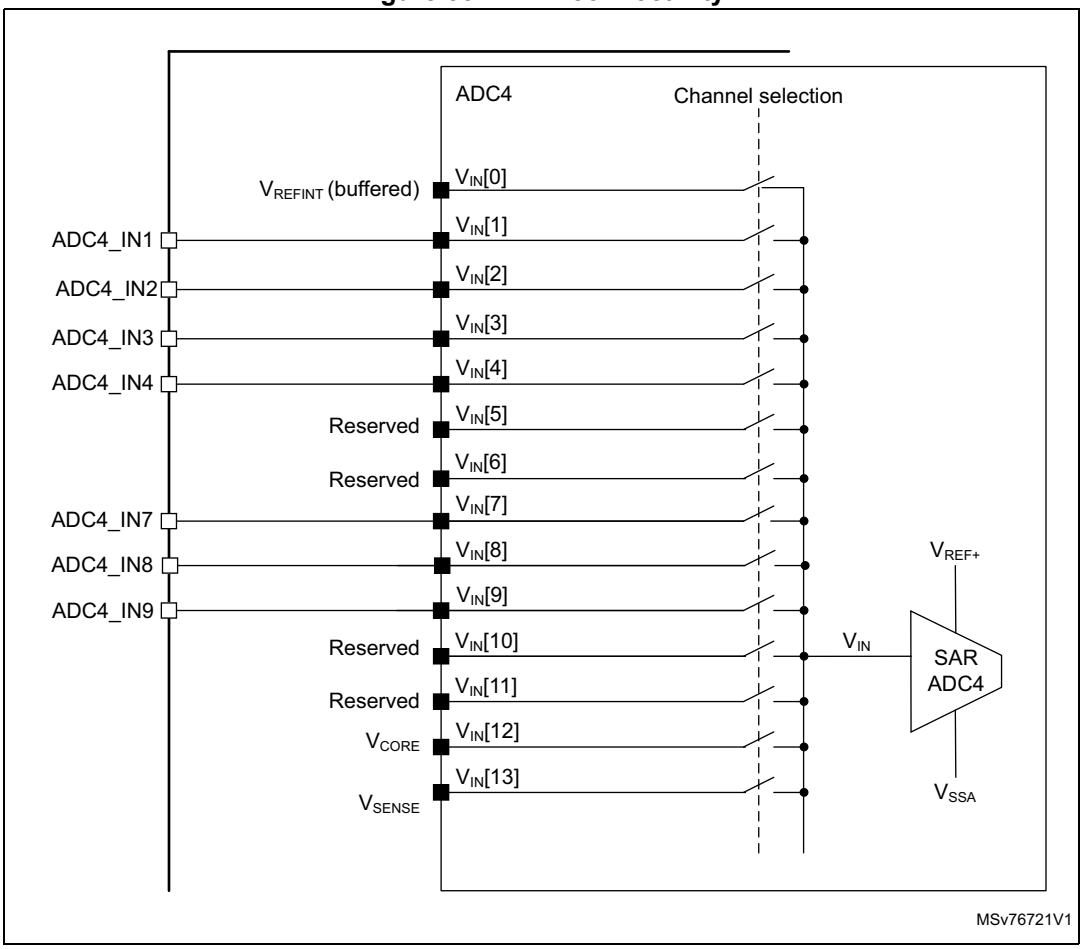
ADC clock source	Latency between the trigger event and the start of conversion
Bus clock divided by 4	Latency is deterministic (no jitter) and equal to 3.75 ADC clock cycles
Bus clock divided by 1	Latency is deterministic (no jitter) and equal to 4 ADC clock cycles

1. Refer to the device datasheet for the maximum F_{ADC} frequency.

22.4.7 ADC connectivity

ADC inputs are connected to the external channels as well as internal sources as described in *Figure 85*.

Figure 85. ADC4 connectivity



22.4.8 Configuring the ADC

The software can write to the ADCAL and ADEN bits in the ADC_CR and ADC_PWRR register if the ADC is disabled (ADEN must be 0).

The software must only write to the ADSTART and ADDIS bits in the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN = 1 and ADDIS = 0).

For all the other control bits in the ADC_IER, ADC_CFGRi, ADC_SMPR, ADC_CHSELR and ADC_CCR registers, refer to the description of the corresponding control bit in *Section 22.7: ADC registers*. If the ADC operates in software trigger mode, set the ADSTP bit in ADC_CR register, then wait until ADSTP bit become 0 before reconfiguring the above registers.

ADC_AWDTRi registers can be modified when a conversion is ongoing.

The software must only write to the ADSTP bit in the ADC_CR register if the ADC is enabled (and possibly converting) and there is no pending request to disable the ADC (ADSTART = 1 and ADDIS = 0).

Note: There is no hardware protection preventing software from making write operations forbidden by the above rules. If such a forbidden write access occurs, the ADC may enter an undefined state. To recover correct operation in this case, the ADC must be disabled (clear ADEN = 0 and all the bits in the ADC_CR register).

22.4.9 Channel selection (CHSEL, SCANDIR, CHSELRMOD)

There are up to 10 multiplexed channels:

- 7 analog inputs from GPIO pins (ADC_INx)
- 3 internal analog inputs: temperature Sensor, internal reference voltage, V_{CORE}

It is possible to convert a single channel or a sequence of channels.

The sequence of the channels to be converted can be programmed in the ADC_CHSELR channel selection register: each analog input channel has a dedicated selection bit (CHSELx).

The ADC scan sequencer can be used in two different modes:

- Sequencer not fully configurable:

  The order in which the channels are scanned is defined by the channel number (CHSELRMOD bit must be cleared in ADC_CFGR1 register):

  – Sequence length configured through CHSELx bits in ADC_CHSELR register

  – Sequence direction: the channels are scanned in a forward direction (from the lowest to the highest channel number) or backward direction (from the highest to the lowest channel number) depending on the value of SCANDIR bit (SCANDIR = 0: forward scan, SCANDIR = 1: backward scan)

  - Any channel can belong to in these sequences
- Fully-configurable sequencer

  The CHSELRMOD bit is set in ADC_CFGR1 register.
  - Sequencer length is up to eight channels
  - The order in which the channels are scanned is independent from the channel number. Any order can be configured through SQ1[3:0] to SQ8[3:0] bits in ADC_CHSELR register.
  - If the sequencer detects SQx[3:0] = 0b1111, the following SQx[3:0] registers are ignored.
  - If no 0b1111 is programmed in SQx[3:0], the sequencer scans full eight channels.

The software is allowed to program the CHSEL, SCANDIR and CHSELRMOD bit only when ADSTART bit is cleared in ADC_CR register. This ensures that no conversion is ongoing. If the ADC operated in software trigger mode, set ADSTP bit then wait until ADSTP bit become 0 before reconfiguring these registers. This sequence must be respected even if ADSTART bit is cleared to 0 after the conversion,

Temperature sensor, V_{REFINT}, and V_{CORE} internal channels

The temperature sensor, the internal reference voltage (V_{REFINT}), and V_{CORE} are connected to ADC internal channels. Refer to *Table ADC interconnection* in *Section 22.4.2: ADC pins and internal signals* for details.

22.4.10 Programmable sampling time (SMPx[2:0])

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 capacitor to the input voltage level.

Having a programmable sampling time allows the conversion speed to be trimmed according to the input resistance of the input voltage source.

The ADC samples the input voltage for a number of ADC clock cycles that can be modified using the SMP1[2:0] and SMP2[2:0] bits in the ADC_SMPR register.

Each channel can choose one out of two sampling times configured in SMP1[2:0] and SMP2[2:0] bitfields, through SMPSELx bits in ADC_SMPR register.

The total conversion time is calculated as follows:

t_{CONV} = Sampling time + 12.5 x ADC clock cycles

Example:

With ADC_CLK = 16 MHz and a sampling time of 1.5 ADC clock cycles:

t_{CONV} = 1.5 + 12.5 = 14 ADC clock cycles = 0.875 µs

The ADC indicates the end of the sampling phase by setting the EOSMP flag.

I/O analog switch voltage booster

The resistance of the I/O analog switch increases when the V_{DDA} voltage is too low. The sampling time must consequently be adapted accordingly (refer to the device datasheet for the corresponding electrical characteristics). This resistance can be minimized at low V_{DDA} voltage by enabling an internal voltage booster through the BOOSTEN bit of the

SYSCFG_CFGR1 register or by selecting a V_{DD} booster voltage through the ANASWVDD bit of the SYSCFG_CFGR1 register.

22.4.11 Single conversion mode (CONT = 0)

In single conversion mode, the ADC performs a single sequence of conversions, converting all the channels once. This mode is selected when CONT is cleared in the ADC_CFGR1 register. Conversion is started by either:

- Setting the ADSTART bit in the ADC_CR register
- Hardware trigger event

Inside the sequence, after each conversion is complete:

- The converted data are stored in the 16-bit ADC_DR register
- 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 EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then the ADC stops until a new external trigger event occurs or the ADSTART bit is set again.

Note: To convert a single channel, program a sequence with a length of 1.

22.4.12 Continuous conversion mode (CONT = 1)

In continuous conversion mode, when a software or hardware trigger event occurs, the ADC performs a sequence of conversions, converting all the channels once and then automatically re-starts and continuously performs the same sequence of conversions. This mode is selected when CONT is set to 1 in the ADC_CFGR1 register. Conversion is started by either:

- Setting the ADSTART bit in the ADC_CR register
- Hardware trigger event

Inside the sequence, after each conversion is complete:

- The converted data are stored in the 16-bit ADC_DR register
- 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 EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

Note: To convert a single channel, program a sequence with a length of 1.

It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.

22.4.13 Starting conversions (ADSTART)

Software starts ADC conversions by setting ADSTART to 1.

When ADSTART is set, the conversion:

- Starts immediately if EXTEN = 00 (software trigger)
- At the next active edge of the selected hardware trigger if EXTEN ≠ 00

The ADSTART bit is also used to indicate whether an ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART remains at 0, indicating that the ADC is idle.

The ADSTART bit is cleared by hardware:

- In single mode with software trigger (CONT = 0, EXTEN = 00)
  - At any end of conversion sequence (EOS = 1)
- In discontinuous mode with software trigger (CONT = 0, DISCEN = 1, EXTEN = 00)
  - At end of conversion (EOC = 1)
- In all cases (CONT = x, EXTEN = XX)
  - After execution of the ADSTP procedure invoked by software (see *Section 22.4.15: Stopping an ongoing conversion (ADSTP)*).

When the ADC operates in autonomous mode (DPD bit transition from 1 to 0, see *Autonomous mode (AUTOFF, DPD)*), the ADSTART bit can be set only when the ADC is powered on. (both LDORDY = 1 and ADRDY = 1). In continuous mode (CONT = 1), the ADSTART bit is not cleared by hardware when the EOS flag is set because the sequence is automatically relaunched.

Note: When hardware trigger is selected in single mode (CONT = 0 and EXTEN = 01), ADSTART is not cleared by hardware when the EOS flag is set. This avoids the need for software having to set the ADSTART bit again and ensures the next trigger event is not missed.

It is necessary to set ADSTP to 1 and wait until ADSTP is cleared before reconfiguring or disabling the ADC, even if ADSTART bit is cleared to after the software triggered ADC conversion mode.

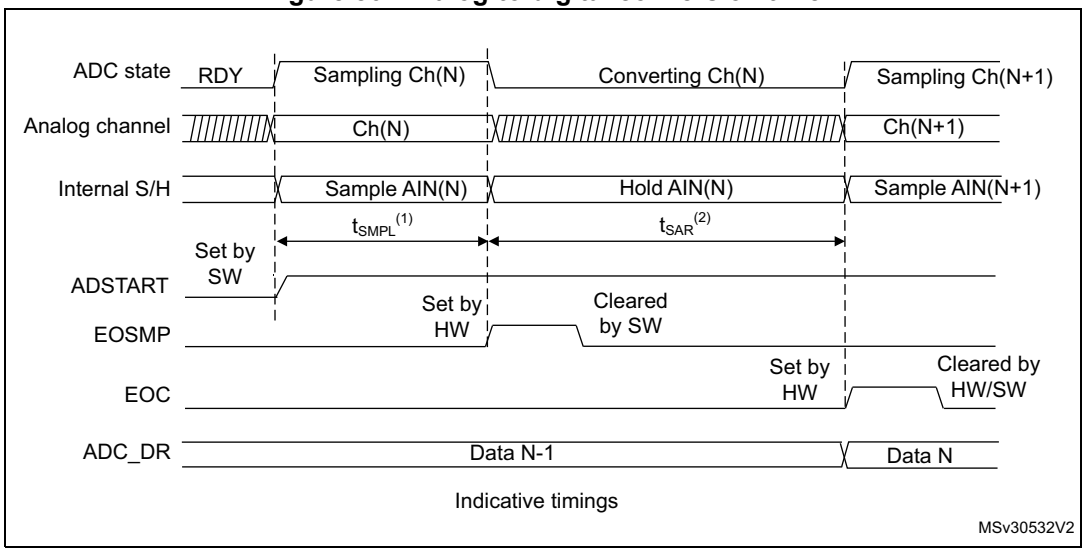
22.4.14 Timings

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

$t_{CONV} = t_{SMPL} + t_{SAR} = [1.5\ _{|min} + 12.5\ _{|12bit}] \times t_{ADC_CLK}$

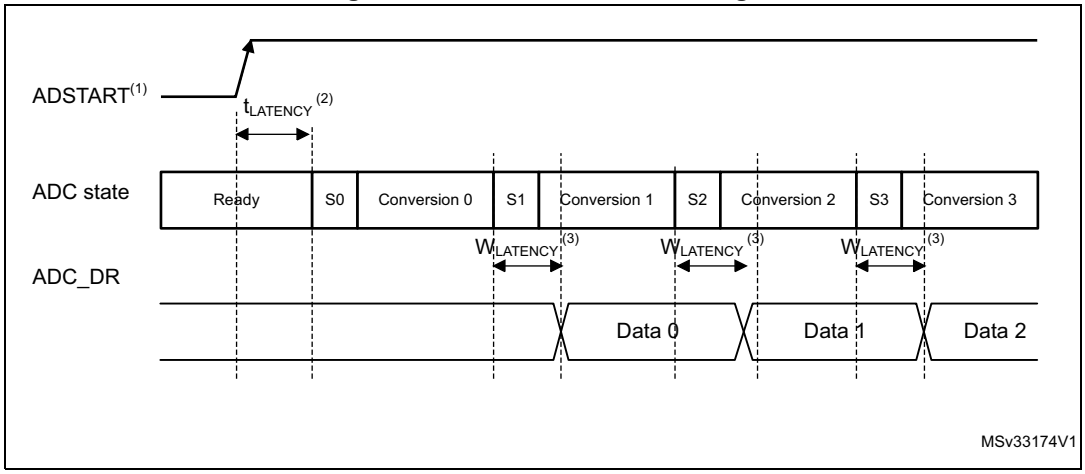
$t_{CONV} = t_{SMPL} + t_{SAR} = 42.9\ ns\ _{|min} + 357.1\ ns\ _{|12bit} = 0.400\ \mu s\ _{|min}$ (for $f_{ADC_CLK} = 35$ MHz)

Figure 86. Analog-to-digital conversion time



1. t_{SMPL} depends on SMP[2:0].
2. t_{SAR} depends on RES[2:0].

Figure 87. ADC conversion timings



1. EXTEN = 00 or EXTEN ≠ 00.
2. Trigger latency (refer to datasheet for more details).
3. ADC_DR register write latency (refer to datasheet for more details).

22.4.15 Stopping an ongoing conversion (ADSTP)

The software can decide to stop any ongoing conversions by setting ADSTP to 1 in the ADC_CR register.

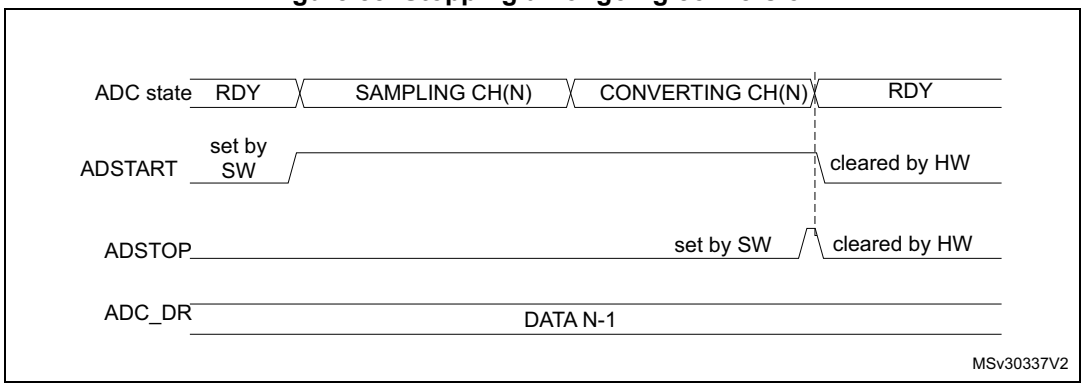
This resets the ADC operation and the ADC is idle, ready for a new operation.

When the ADSTP bit is set by software, any ongoing conversion is aborted and the result is discarded (ADC_DR register is not updated with the current conversion).

The scan sequence is also aborted and reset (meaning that restarting the ADC would re-start a new sequence).

Once this procedure is complete, the ADSTP and ADSTART bits are both cleared by hardware and the software must wait until ADSTART is cleared to 0 before starting new conversions.

Figure 88. Stopping an ongoing conversion



22.4.16 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN)

A conversion or a sequence of conversion can be triggered either by software or by an external event (for example timer capture). If the EXTEN[1:0] control bits are not equal to “0b00”, then external events are able to trigger a conversion with the selected polarity. The trigger selection is effective once software has set bit ADSTART to 1.

Any hardware triggers which occur while a conversion is ongoing are ignored.

If bit ADSTART is cleared, any hardware triggers which occur are ignored.

Table 169 provides the correspondence between the EXTEN[1:0] values and the trigger polarity.

Table 169. Configuring the trigger polarity

Source	EXTEN[1:0]
Trigger detection disabled	00
Detection on rising edge	01
Detection on falling edge	10
Detection on both rising and falling edges	11

Note: The polarity of the external trigger can be changed only when the ADC is not converting (ADSTART = 0).

The EXTSEL[2:0] control bits are used to select which of 8 possible events can trigger conversions.

Refer to Table ADC interconnection in *Section 22.4.2: ADC pins and internal signals* for the list of all the external triggers that can be used for regular conversion.

The software source trigger events can be generated by setting the ADSTART bit in the ADC_CR register.

Note: The trigger selection can be changed only when the ADC is not converting (ADSTART = 0).

22.4.17 Discontinuous mode (DISCEN)

This mode is enabled by setting the DISCEN bit in the ADC_CFGR1 register.

In this mode (DISCEN = 1), a hardware or software trigger event is required to start each conversion defined in the sequence. On the contrary, if DISCEN is cleared, a single hardware or software trigger event successively starts all the conversions defined in the sequence.

Example:

- DISCEN = 1, channels to be converted are channels 0, 3, 7 and 10
  - 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 EOS 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
  - ...
- DISCEN = 0, channels to be converted are channels 0, 3, 7 and 10
  - 1st trigger: the complete sequence is converted: channel 0, then 3, 7 and 10. Each conversion generates an EOC event and the last one also generates an EOS event.
  - Any subsequent trigger events restarts the complete sequence.

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.

22.4.18 Programmable resolution (RES) - fast conversion mode

It is possible to obtain faster conversion times (t_{SAR}) by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the RES[1:0] bits in the ADC_CFGR1 register. Lower resolution allows faster conversion times for applications where high data precision is not required.

Note: The RES[1:0] bit must only be changed when the ADEN bit is reset.

The result of the conversion is always 12 bits wide and any unused LSB bits are read as zeros.

Lower resolution reduces the conversion time needed for the successive approximation steps as shown in *Table 170*.

Table 170. t_{SAR} timings depending on resolution

RES[1:0] bits	t_{SAR} (ADC clock cycles)	t_{SAR} (ns) at f_{ADC} = 35 MHz	$t_{SMPL\ (min)}$ (ADC clock cycles)	t_{CONV} (ADC clock cycles) (with min. t_{SMPL})	t_{CONV} (ns) at f_{ADC} = 35 MHz
12	12.5	357	1.5	14	400
10	10.5	300	1.5	12	343
8	8.5	243	1.5	10	286
6	6.5	186	1.5	8	229

22.4.19 End of conversion, end of sampling phase (EOC, EOSMP flags)

The ADC indicates each end of conversion (EOC) event.

The ADC sets the EOC flag in the ADC_ISR register as soon as a new conversion data result is available in the ADC_DR register. An interrupt can be generated if the EOCIE bit is set in the ADC_IER register. The EOC flag is cleared by software either by writing 1 to it, or by reading the ADC_DR register.

The ADC also indicates the end of sampling phase by setting the EOSMP flag in the ADC_ISR register. The EOSMP flag is cleared by software by writing1 to it. An interrupt can be generated if the EOSMPIE bit is set in the ADC_IER register.

The aim of this interrupt is to allow the processing to be synchronized with the conversions. Typically, an analog multiplexer can be accessed in hidden time during the conversion phase, so that the multiplexer is positioned when the next sampling starts.

Note: As there is only a very short time left between the end of the sampling and the end of the conversion, it is recommenced to use polling or a WFE instruction rather than an interrupt and a WFI instruction.

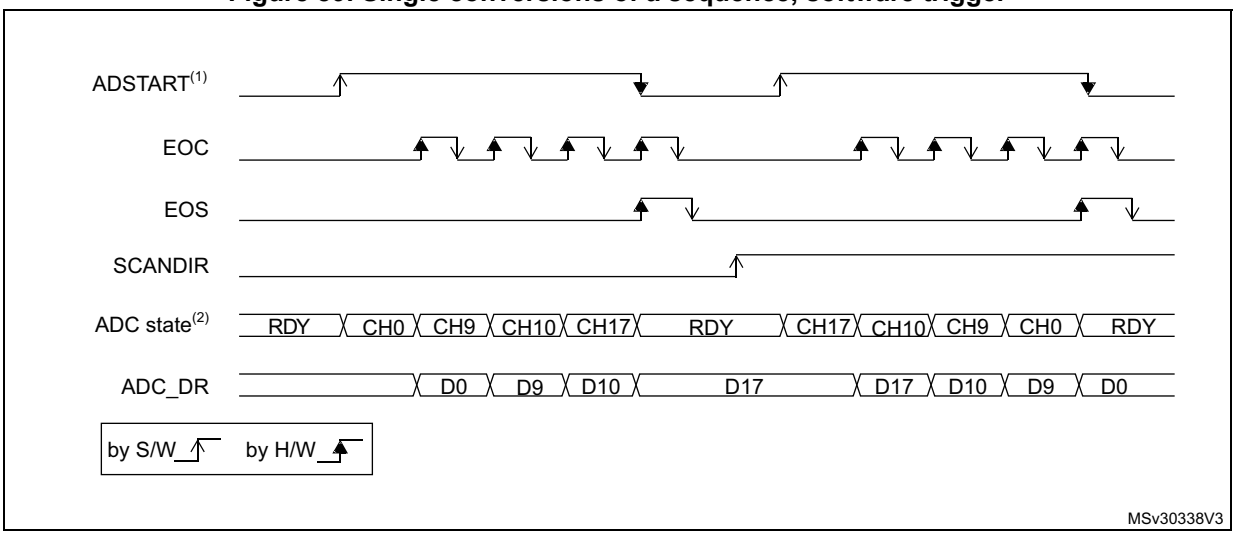
22.4.20 End of conversion sequence (EOS flag)

The ADC notifies the application of each end of sequence (EOS) event.

The ADC sets the EOS flag in the ADC_ISR register as soon as the last data result of a conversion sequence is available in the ADC_DR register. An interrupt can be generated if the EOSIE bit is set in the ADC_IER register. The EOS flag is cleared by software by writing 1 to it.

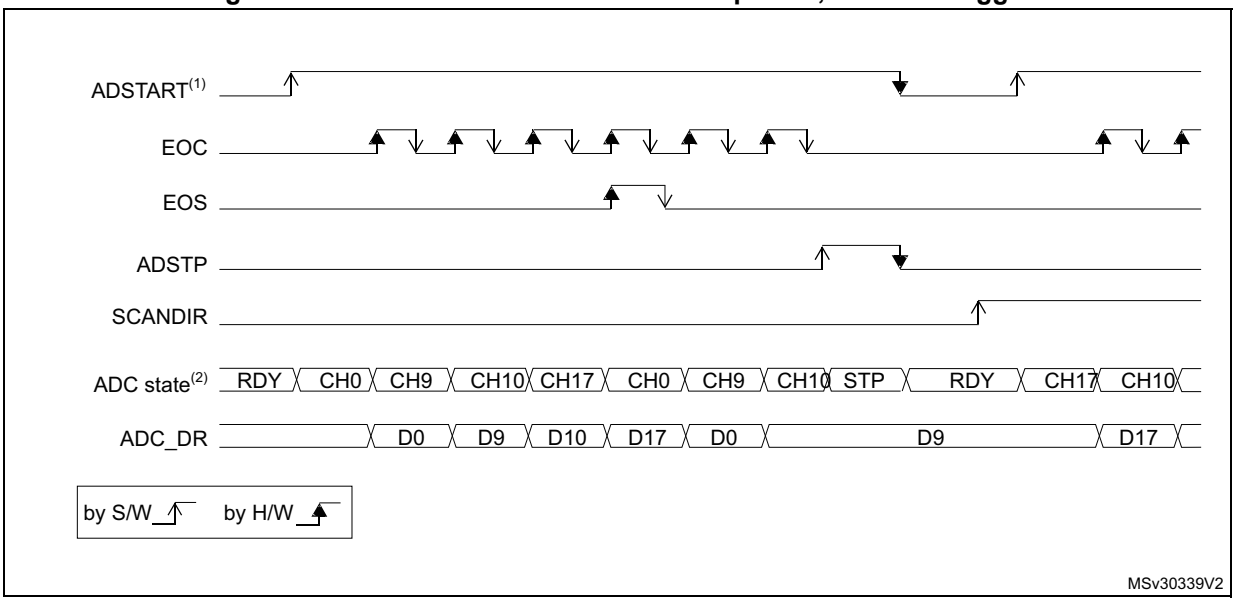
22.4.21 Example timing diagrams (single/continuous modes hardware/software triggers)

Figure 89. Single conversions of a sequence, software trigger



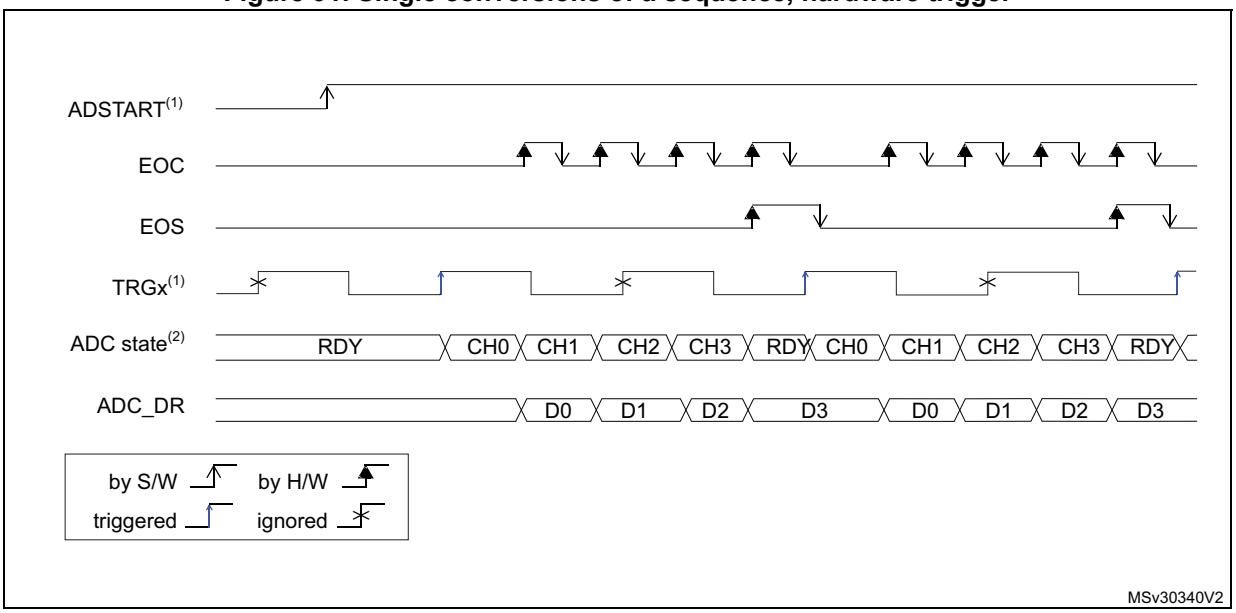
1. EXTEN = 00, CONT = 0.
2. CHSEL = 0x20601, WAIT = 0, AUTOFF = 0.

Figure 90. Continuous conversion of a sequence, software trigger



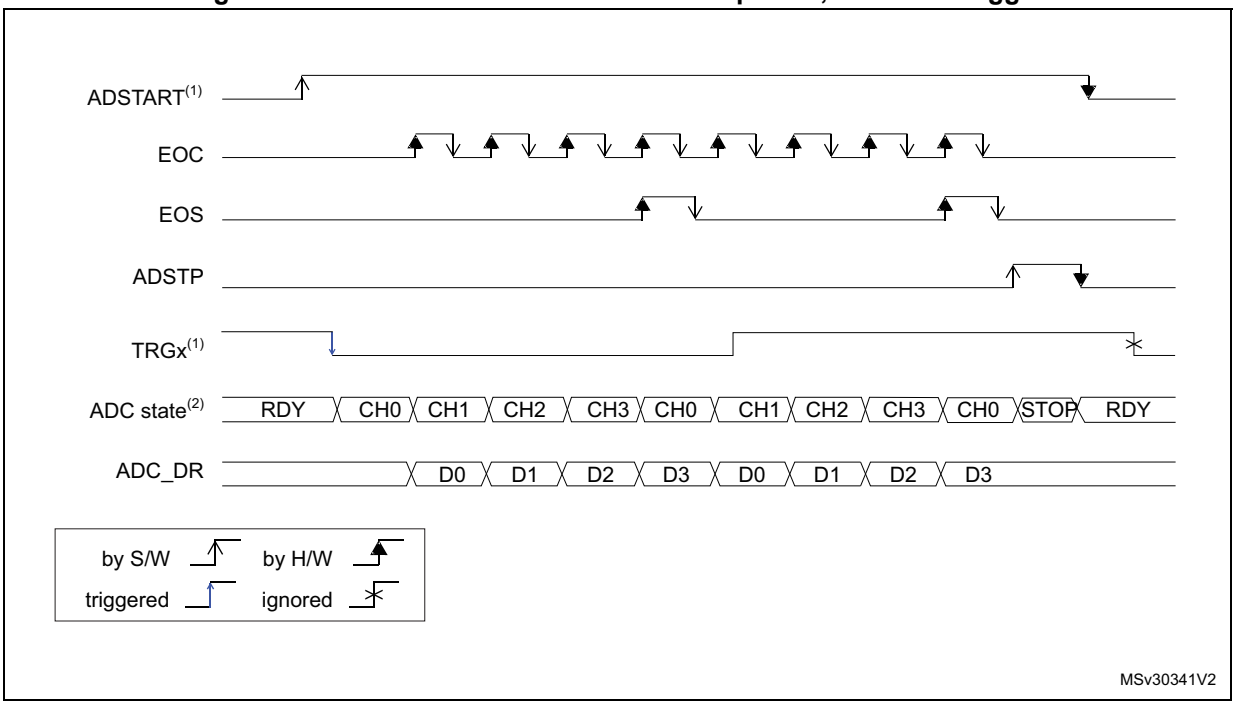
1. EXTEN = 00, CONT = 1.
2. CHSEL = 0x20601, WAIT = 0, AUTOFF = 0.

Figure 91. Single conversions of a sequence, hardware trigger



1. EXTSEL = TRGx (over-frequency), EXTEN = 01 (rising edge), CONT = 0.
2. CHSEL = 0xF, SCANDIR = 0, WAIT = 0, AUTOFF = 0.

Figure 92. Continuous conversions of a sequence, hardware trigger



1. EXTSEL = TRGx, EXTEN = 10 (falling edge), CONT = 1.
2. CHSEL = 0xF, SCANDIR = 0, WAIT = 0, AUTOFF = 0.

22.4.22 Low-frequency trigger mode

If the application has to support a time longer than the maximum t_{IDLE} value (between one trigger to another for single conversion mode or between the ADC enable and the first ADC conversion), then the ADC internal state needs to be rearmed. This mechanism can be enabled by setting LFTRIG bit to 1 in ADC_CFGR2 register. By setting this bit, any trigger (software or hardware) sends a rearm command to ADC. The conversion is started after a two ADC clock cycle delay compared to LFTRIG set to 0.

It is not necessary to use this mode when AUTOFF bit is set to 1. For wait mode, only the first trigger generates an internal rearm command.

22.4.23 Data management

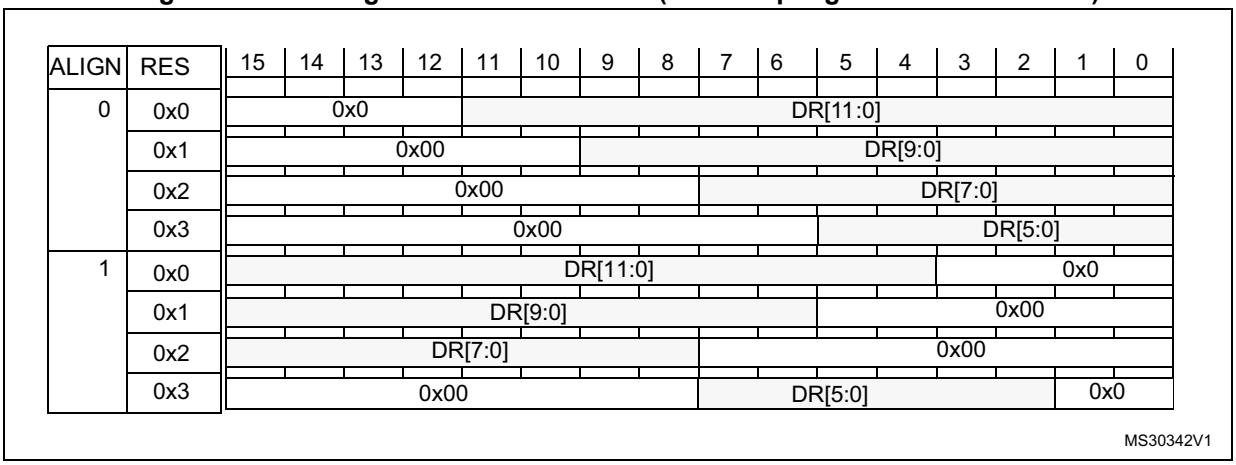
Data register and data alignment (ADC_DR, ALIGN)

At the end of each conversion (when an EOC event occurs), the result of the converted data is stored in the ADC_DR data register which is 16-bit wide.

The format of the ADC_DR depends on the configured data alignment and resolution.

The ALIGN bit in the ADC_CFGR1 register selects the alignment of the data stored after conversion. Data can be right-aligned (ALIGN = 0) or left-aligned (ALIGN = 1) as shown in *Figure 93*.

Figure 93. Data alignment and resolution (oversampling disabled: OVSE = 0)



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 data from a new conversion is available.

The OVR flag is set in the ADC_ISR register if the EOC 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_IER register.

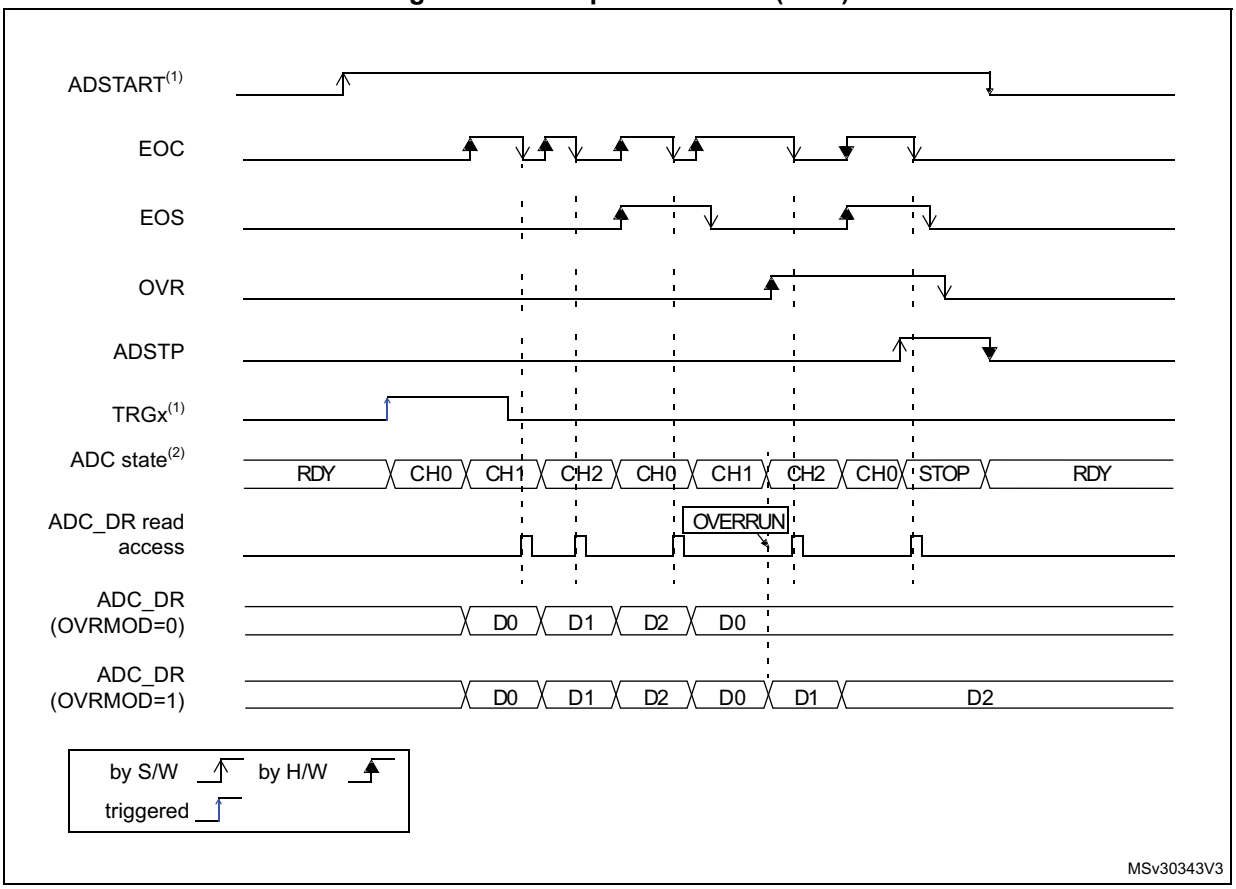
When an overrun condition occurs, the ADC keeps operating and can continue to convert unless the software decides to stop and reset the sequence by setting the ADSTP bit in the ADC_CR register.

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_CFGR1 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 and the previous unread data is lost. If OVR remains at 1, further conversions can be performed and the ADC_DR register always contains the data from the latest conversion.

Figure 94. Example of overrun (OVR)



Managing a sequence of data converted without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by software. In this case the software must use the EOC flag and its associated interrupt to handle each data result. Each time a conversion is complete, the EOC bit is set in the ADC_ISR register and the ADC_DR register can be read. The OVRMOD bit in the ADC_CFGR1 register must be configured to 0 to manage overrun events as an error.

Managing converted data without using the DMA without overrun

It may be useful to let the ADC convert one or more channels without reading the data after each conversion. In this case, the OVRMOD bit must be configured at 1 and the OVR flag must be ignored by the software. When OVRMOD is et to 1, an overrun event does not prevent the ADC from continuing to convert and the ADC_DR register always contains the latest conversion data.

Managing converted data using the DMA

Since all converted channel values are stored in a single data register, it is efficient to use DMA when converting more than one channel. This avoids losing the conversion data results stored in the ADC_DR register.

When DMA mode is enabled (DMAEN bit set to 1 in the ADC_CFGR1 register), a DMA request is generated after the conversion of each channel. This allows the transfer of the

converted data from the ADC_DR register to the destination location selected by the software.

Note: The DMAEN bit in the ADC_CFGR1 register must be set after the ADC calibration phase.

Despite this, if an overrun occurs (OVR = 1) because the DMA did not serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA. Which means that all the data transferred to the RAM can be considered as valid.

Depending on the configuration of OVRMOD bit, the data is either preserved or overwritten (refer to *ADC overrun (OVR, OVRMOD)*).

The DMA transfer requests are blocked until the software clears the OVR bit.

Two different DMA modes are proposed depending on the application use and are configured with bit DMACFG in the ADC_CFGR1 register:

- DMA one-shot mode (DMACFG = 0).

  This mode must be selected when the DMA is programmed to transfer a fixed number of data words.
- DMA circular mode (DMACFG = 1)

  This mode must be selected when programming the DMA in circular mode or double buffer mode.

DMA one-shot mode (DMACFG = 0)

In this mode, the ADC generates a DMA transfer request each time a new conversion data word is available and stops generating DMA requests once the DMA has reached the last DMA transfer (when a transfer complete interrupt occurs - refer to DMA section), even if a conversion has been started again.

When the DMA transfer is complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- Any ongoing conversion is aborted and its partial result discarded
- No new DMA request is issued to the DMA controller. This avoids generating an overrun error if there are still conversions which are started.
- The scan sequence is stopped and reset
- The DMA is stopped

DMA circular mode (DMACFG = 1)

In this mode, the ADC generates a DMA transfer request each time a new conversion data word is available in the data register, even if the DMA has reached the last DMA transfer. This allows the DMA configuration in circular mode in order to handle a continuous analog input data stream.

22.4.24 Low-power features

Wait conversion mode (WAIT)

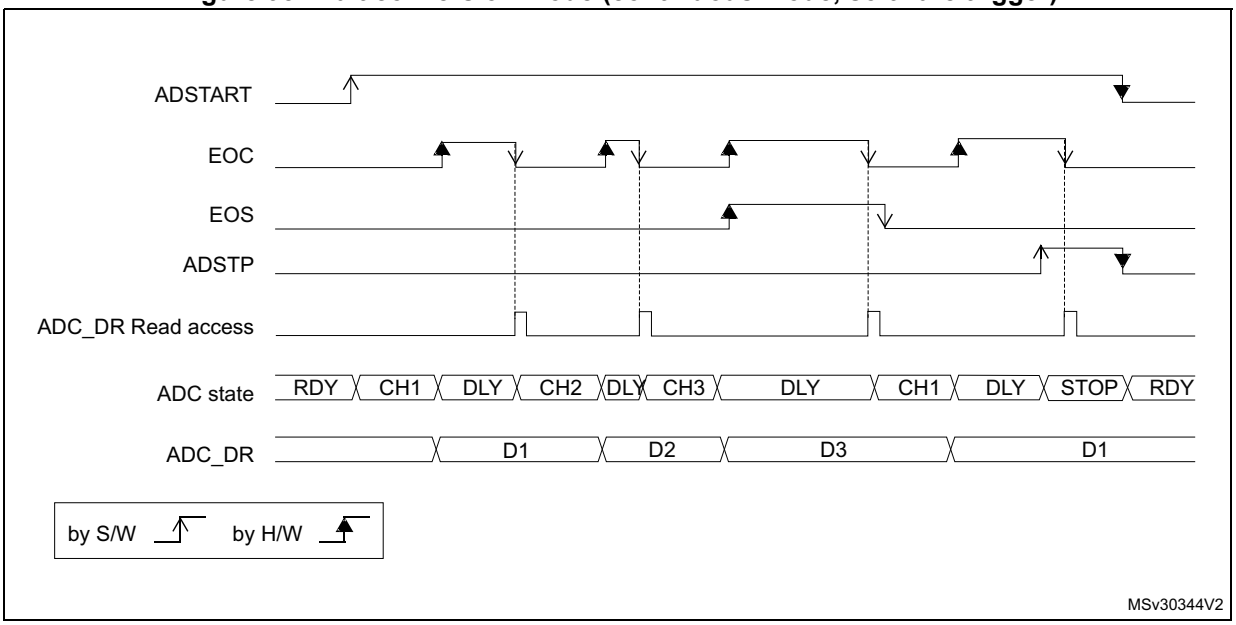
Wait conversion mode can be used to simplify the software as well as optimizing the performance of applications clocked at low frequency where there might be a risk of ADC overrun occurring.

When the WAIT bit is set to 1 in the ADC_CFGR1 register, a new conversion can start only if the previous data has been treated, once the ADC_DR register has been read or if the EOC bit has been cleared.

This is a way to automatically adapt the speed of the ADC to the speed of the system that reads the data.

Note: Any hardware triggers which occur while a conversion is ongoing or during the wait time preceding the read access are ignored.

Figure 95. Wait conversion mode (continuous mode, software trigger)



1. EXTEN = 00, CONT = 1.
2. CHSEL = 0x3, SCANDIR = 0, WAIT = 1, AUTOFF = 0.

ADC power-saving modes

The ADC embeds two power-saving modes, the auto-off and the autonomous modes.

Auto-off mode (AUTOFF)

The auto-off mode is enabled by setting the AUTOFF bit to 1 in the ADC_PWRR register.

Below the auto-off mode operating sequence:

1. When AUTOFF is set to 1, the ADC is always powered off when no conversion is ongoing.
2. It then automatically wakes up when a conversion is triggered by software or by hardware, and a startup time is inserted between the trigger event and the ADC sampling time.
3. The ADC is then automatically disabled once the conversion or sequence of conversions is complete.
4. When consecutive hardware or software triggers occur, the ADC is automatically enabled and the conversion is processed.

Refer to *Figure 96* for a description of auto-off mode state diagram.

The auto-off mode dramatically reduces power consumption in applications requiring a limited number of conversions or conversion requests far between enough (for example with a low-frequency hardware trigger) to justify the extra power and time used for switching the ADC on and off.

Auto-off mode can be combined with wait mode (WAIT = 1) for applications clocked at low frequency. This combination can achieve significant power saving if the ADC is automatically powered off during the wait phase and restarted as soon as the ADC_DR register is read by the application (see *Figure 97: ADC behavior with WAIT = 0 and AUTOFF = 1* and *Figure 98: ADC behavior with WAIT = 1 and AUTOFF = 1*).

The auto-off mode is compatible with the autonomous peripheral mode.

Note: Refer to the Section Reset and clock control (RCC) for the description of how to manage the dedicated internal oscillators. The ADC interface can automatically switch on/off these internal oscillators to save power.

Figure 96. Auto-off mode state diagram

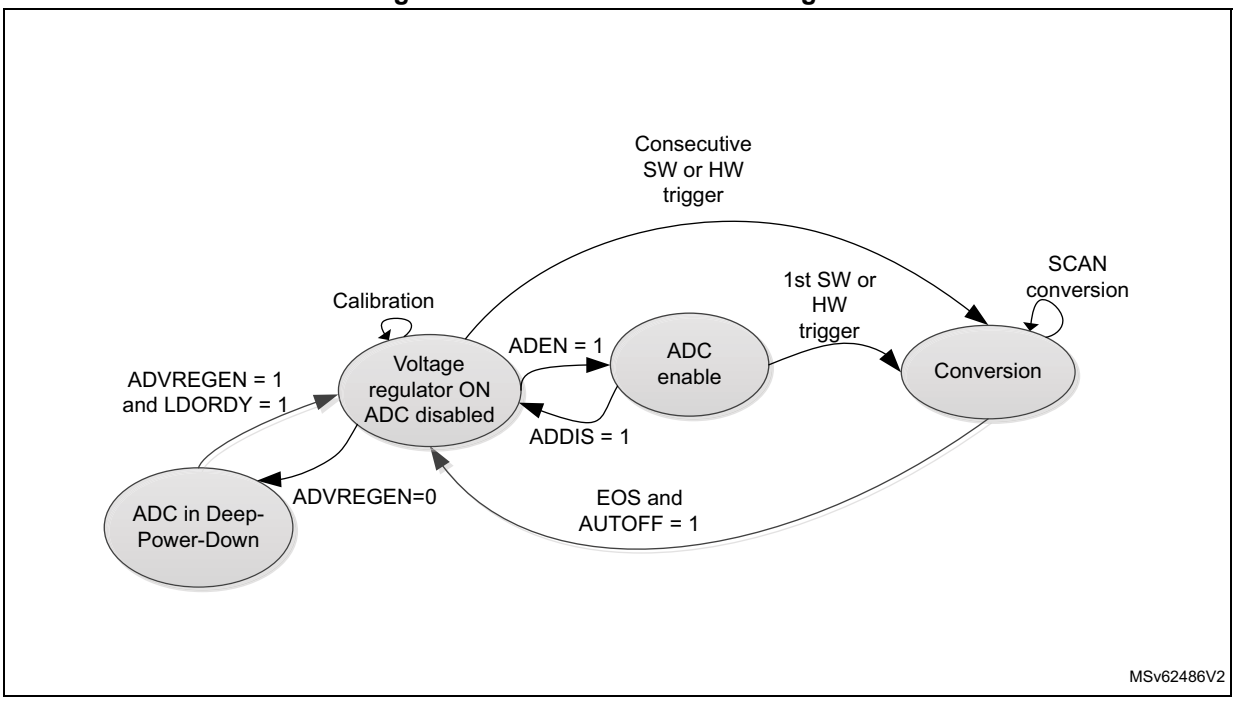
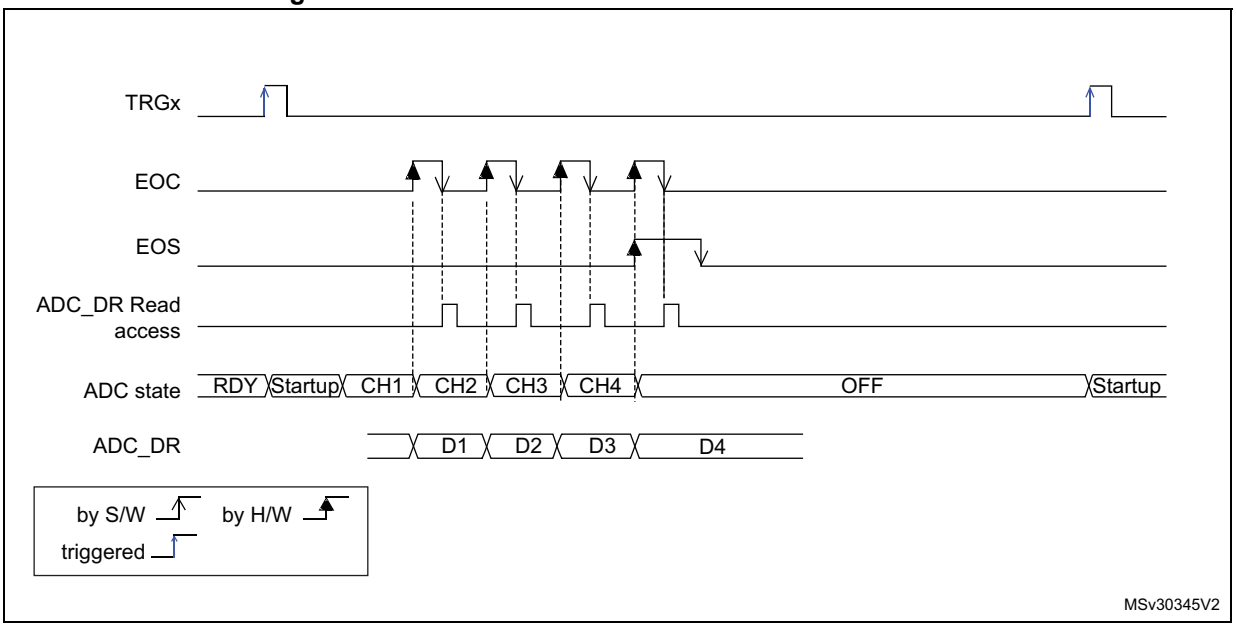
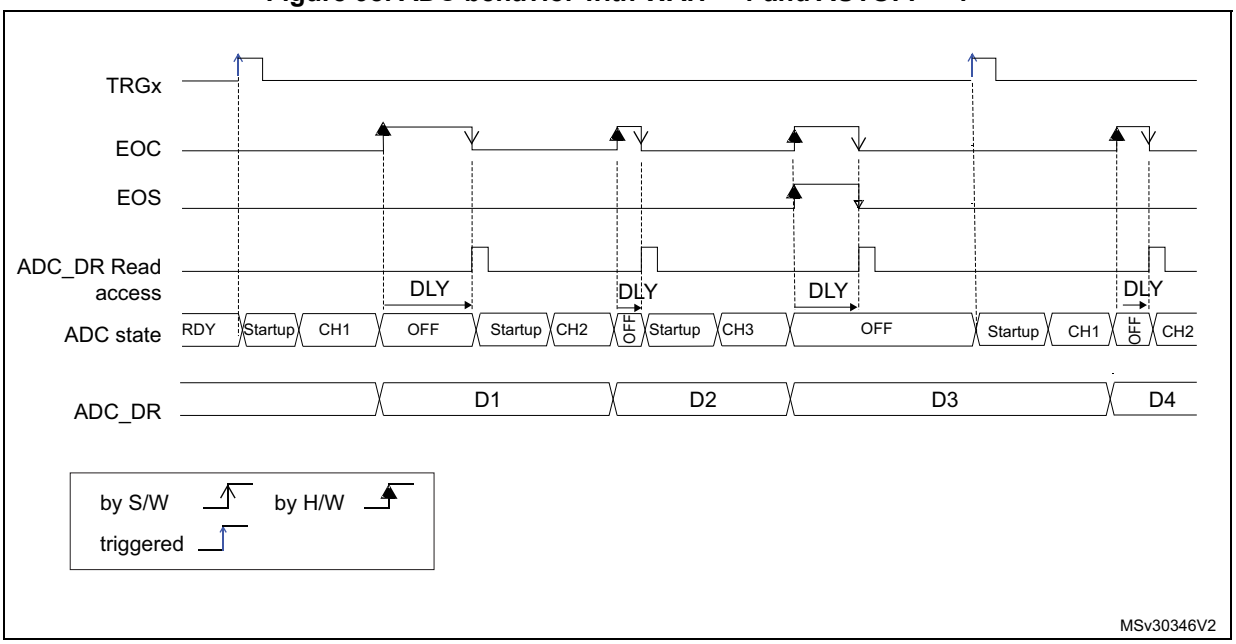


Figure 97. ADC behavior with WAIT = 0 and AUTOFF = 1



1. EXTSEL = TRGx, EXTEN = 01 (rising edge), CONT = x, ADSTART = 1, CHSEL = 0xF, SCANDIR = 0, WAIT = 0, AUTOFF = 1.

Figure 98. ADC behavior with WAIT = 1 and AUTOFF = 1



1. EXTSEL = TRGx, EXTEN = 01 (rising edge), CONT = x, ADSTART = 1, CHSEL = 0xF, SCANDIR = 0, WAIT = 1, AUTOFF = 1.

Autonomous mode (AUTOFF, DPD)

The autonomous mode is enabled by setting both AUTOFF and DPD bits to 1 in ADC_PWRR register. In addition, the autonomous mode must be enabled in the RCC.

Below the autonomous mode operating sequence:

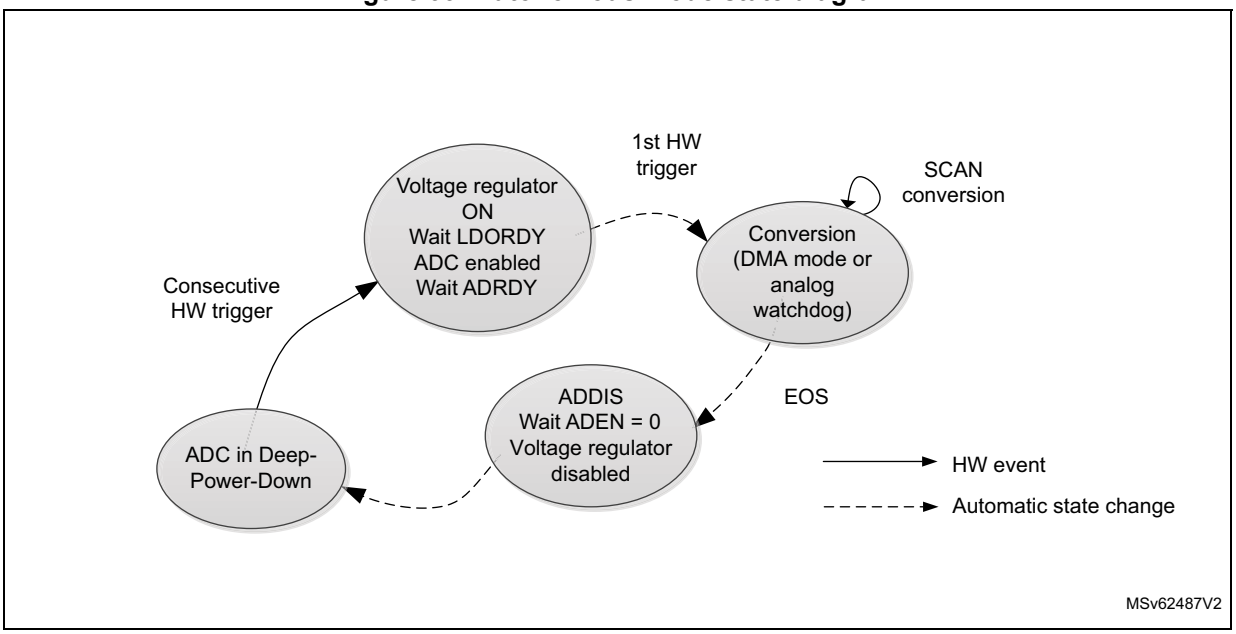
1. When AUTOFF and DPD are both set to 1, the ADC is powered off when no conversion is ongoing.
2. Upon hardware trigger reception, the ADC requests the adc_ker_ck and adc_hclk clocks to the RCC, the ADC voltage regulator is enabled, the calibration factor is loaded, the ADC is enabled and the conversion starts.
3. Once the ADC conversion is complete, the ADC can either generate an AWDx interrupt or a DMA request, depending on peripheral configuration:
   - When DMA mode is enabled, the ADC generates a DMA request to transfer data to memory or to another peripherals.
   - When an analog watchdog is enabled, ADC data do not need to be transferred. The analog watchdog compares the data to the threshold value and generates an AWDx interrupt to wake up the device if the data is under or over the programmed threshold.
4. When the ADC conversion/sequence or conversion is complete, the ADC and the ADC voltage regulator are automatically disabled as well as V_{REFINT} buffer and the temperature sensor, and further clock requests are deasserted. This allows the minimization of current consumption.
5. When consecutive hardware triggers occur, the ADC is automatically enabled and the conversion is processed.

Refer to *Figure 99* for a description of autonomous mode state diagram.

The autonomous mode enables the ADC peripheral to operate when the device is in Stop mode. However it can also be used in Run or Sleep mode.

It is compatible with the autonomous peripheral mode.

Figure 99. Autonomous mode state diagram



22.4.25 Analog window watchdog

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

Description of analog watchdog 1

AWD1 analog watchdog is enabled by setting the AWD1EN bit in the ADC_CFGR1 register. It is used to monitor that either one selected channel or all enabled channels (see *Table 172: Analog watchdog 1 channel selection*) remain within a configured voltage range (window) as shown in *Figure 100*.

The AWD1 analog watchdog 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 HT1[11:0] and LT1[11:0] bits of ADC_AWD1TR register. An interrupt can be enabled by setting the AWD1IE bit in the ADC_IER register.

The AWD1 flag is cleared by software by programing it to 1.

When converting data with a resolution of less than 12-bit (according to bits DRES[1:0]), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned).

Table 171 describes how the comparison is performed for all the possible resolutions.

Table 171. Analog watchdog comparison

Resolution bits RES[1:0]	Analog Watchdog comparison between:		Comments
	Raw converted data, left aligned(1)	Thresholds	
00: 12-bit	DATA[11:0]	LTx[11:0] and HTx[11:0]	-
01: 10-bit	DATA[11:2],00	LTx[11:0] and HTx[11:0]	The user must configure LTx[1:0] and HTx[1:0] to “00”
10: 8-bit	DATA[11:4],0000	LTx[11:0] and HTx[11:0]	The user must configure LTx[3:0] and HTx[3:0] to “0000”
11: 6-bit	DATA[11:6],000000	LTx[11:0] and HTx[11:0]	The user must configure LTx[5:0] and HTx[5:0] to “000000”

1. The watchdog comparison is performed on the raw converted data before any alignment calculation.

Table 172 shows how to configure the AWD1SGL and AWD1EN bits in the ADC_CFGR1 register to enable the analog watchdog on one or more channels.

Figure 100. Analog watchdog guarded area

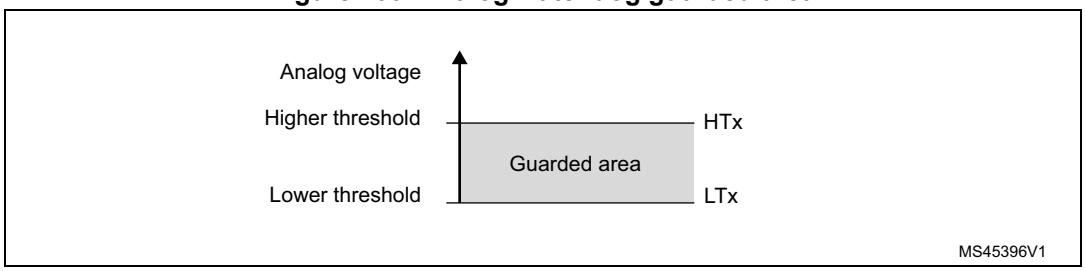


Table 172. Analog watchdog 1 channel selection

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit
None	x	0
All channels	0	1
Single(1) channel	1	1

1. Selected by the AWD1CH[4:0] bits

Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the AWDxCHy in ADC_AWDxCR (x = 2, 3).

The corresponding watchdog is enabled when any AWDxCHy bit (x = 2,3) is set in ADC_AWDxCR register.

When converting data with a resolution of less than 12 bits (configured through DRES[1:0] bits), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned).

Table 171 describes how the comparison is performed for all the possible resolutions.

The AWD2/3 analog watchdog status bit is set if the analog voltage converted by the ADC is below a low threshold or above a high threshold. These thresholds are programmed in

HTx[11:0] and LTx[11:0] of ADC_AWDxTR registers (x = 2 or 3). An interrupt can be enabled by setting the AWDxIE bit in the ADC_IER register.

The AWD2 and AWD3 flags are cleared by software by programming them to 1.

ADC_AWDx_OUT signal output generation

Each analog watchdog is associated to an internal hardware signal, ADC_AWDx_OUT (x being the watchdog number) that is directly connected to the ETR input (external trigger) of some on-chip timers (refer to the timers section for details on how to select the ADC_AWDx_OUT signal as ETR).

ADC_AWDx_OUT is activated when the associated analog watchdog is enabled:

- ADC_AWDx_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADC_AWDx_OUT is reset after the end of the next guarded conversion which is inside the programmed thresholds. It remains at 1 if the next guarded conversions are still outside the programmed thresholds.
- ADC_AWDx_OUT is also reset when disabling the ADC (when setting ADDIS to 1). Note that stopping conversions (ADSTP set to 1), might clear the ADC_AWDx_OUT state.
- ADC_AWDx_OUT state does not change when the ADC converts the none-guarded channel (see *Figure 103*)

AWDx flag is set by hardware and reset by software: AWDx flag has no influence on the generation of ADC_AWDx_OUT (as an example, ADC_AWDx_OUT can toggle while AWDx flag remains at 1 if the software has not cleared the flag).

The ADC_AWDx_OUT signal is generated by the ADC_CLK domain. This signal can be generated even the bus clock is stopped.

The AWD comparison is performed at the end of each ADC conversion. The ADC_AWDx_OUT rising edge and falling edge occurs two ADC_CLK clock cycles after the comparison.

As ADC_AWDx_OUT is generated by the ADC_CLK domain and AWD flag is generated by the bus clock domain, the rising edges of these signals are not synchronized.

Figure 101. ADC_AWDx_OUT signal generation

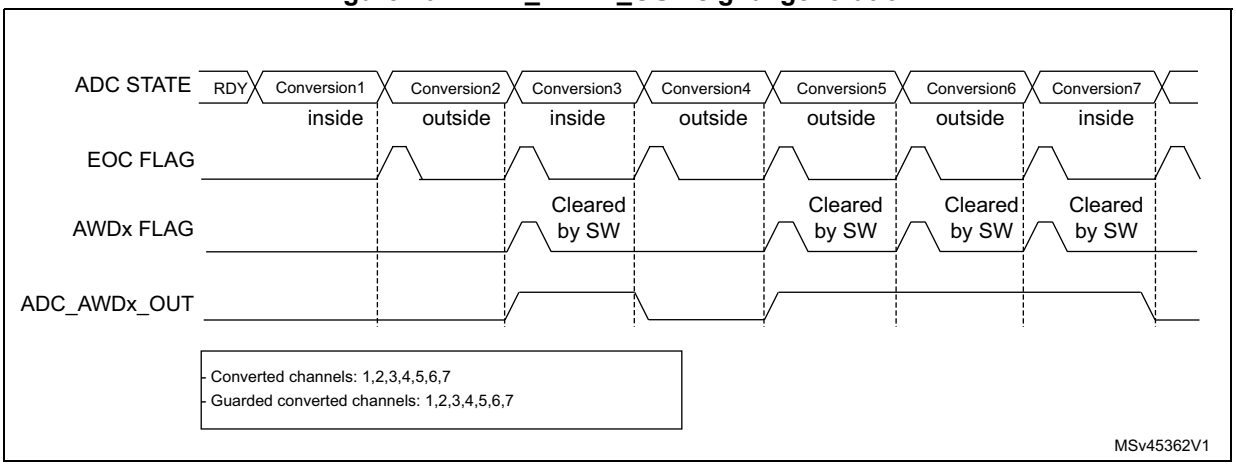


Figure 102. ADC_AWDx_OUT signal generation (AWDx flag not cleared by software)

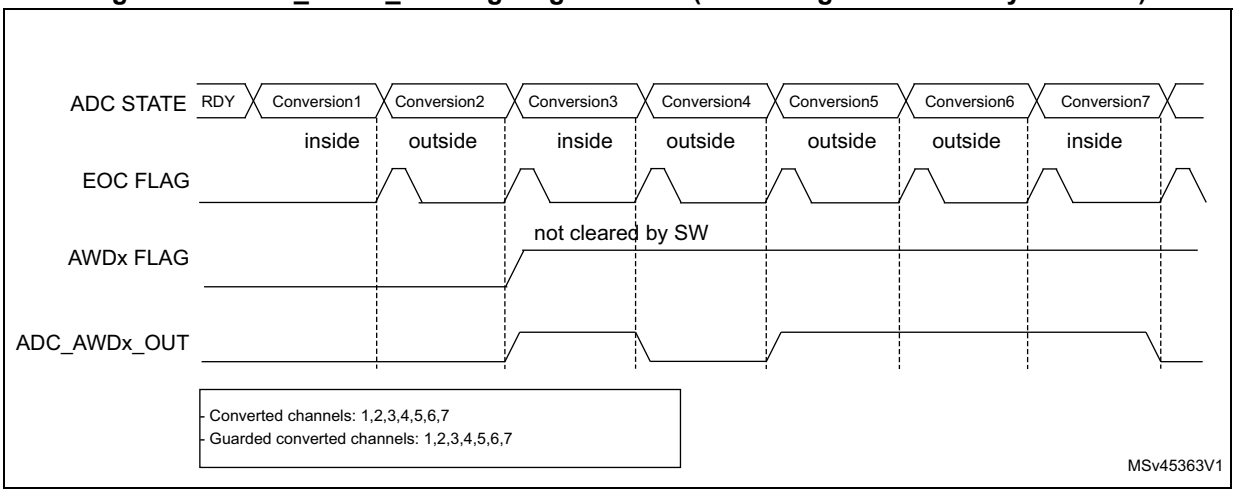
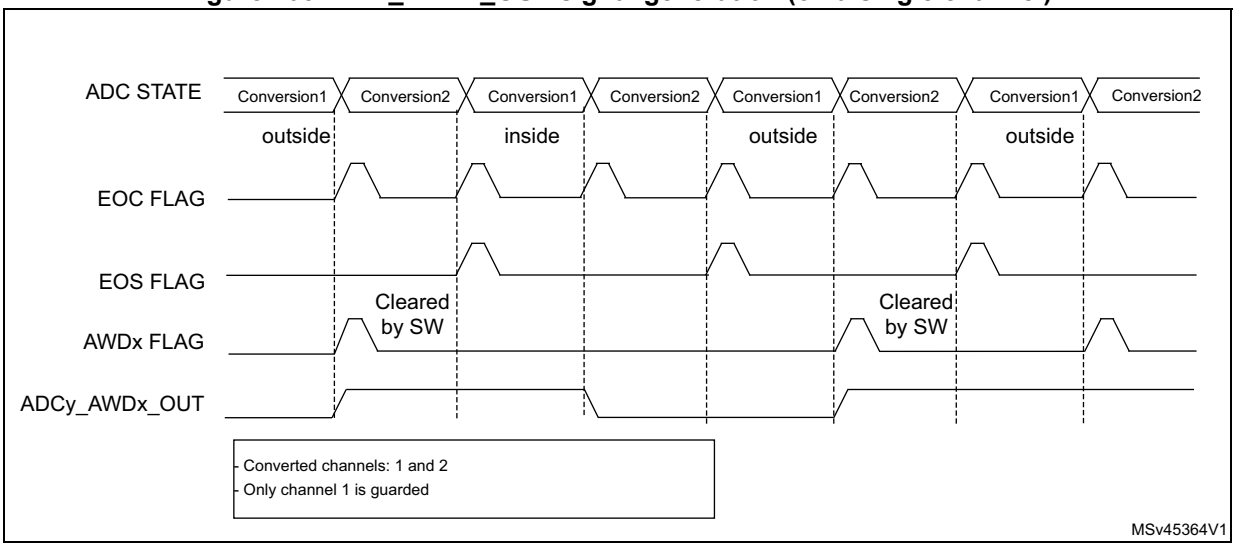


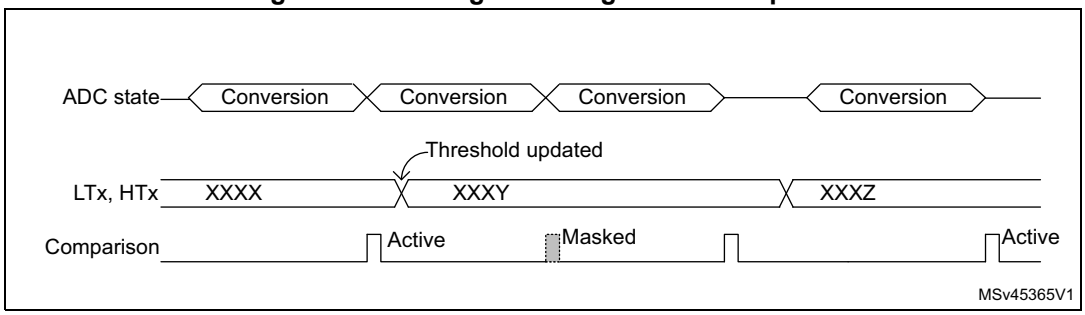
Figure 103. ADC_AWDx_OUT signal generation (on a single channel)



Analog watchdog threshold control

LTx[11:0] and HTx[11:0] can be changed during an analog-to-digital conversion (that is between the start of the conversion and the end of conversion of the ADC internal state). If HTx and LTx bits are programmed during the ADC guarded channel conversion, the watchdog function is masked for this conversion. This mask is cleared when starting a new conversion, and the resulting new AWD threshold is applied starting the next ADC conversion result. AWD comparison is performed at each end of conversion. If the current ADC data are out of the new threshold interval, this does not generated any interrupt or an ADC_AWDx_OUT signal. The Interrupt and the ADC_AWDx_OUT generation only occurs at the end of the ADC conversion that started after the threshold update. If ADC_AWDx_OUT is already asserted, programming the new threshold does not deassert the ADC_AWDx_OUT signal.

Figure 104. Analog watchdog threshold update



22.4.26 Oversampler

The oversampling unit performs data preprocessing to offload the CPU. It can handle multiple conversions and average them into a single data with increased data width, up to 16-bit.

It provides a result with the following form, where N and M can be adjusted:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)$$

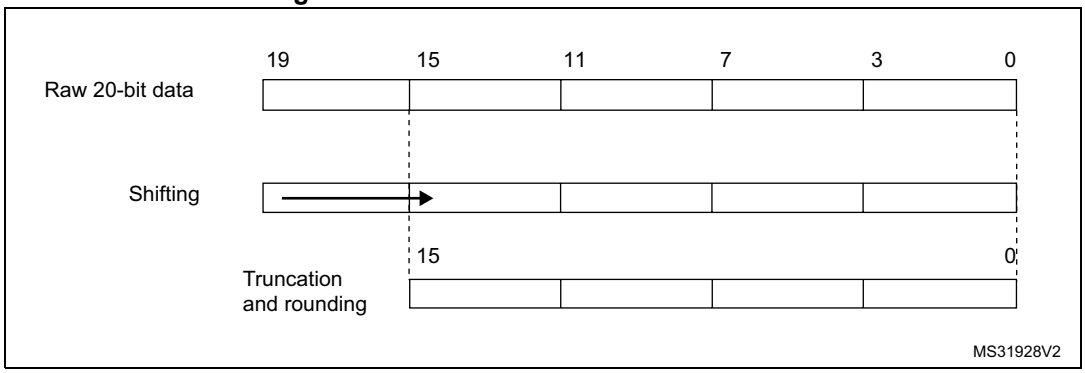
It allows the following functions to be performed by hardware: averaging, data rate reduction, SNR improvement, basic filtering.

The oversampling ratio N is defined using the OVSR[2:0] bits in the ADC_CFGR2 register. It can range from 2x to 256x. The division coefficient M consists of a right bit shift up to 8 bits. It is configured through the OVSS[3:0] bits in the ADC_CFGR2 register.

The summation unit can yield a result up to 20 bits (256 x 12-bit), which is first shifted right. The lower bits of the result are then truncated, keeping only the 16 least significant bits rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred into the ADC_DR data register.

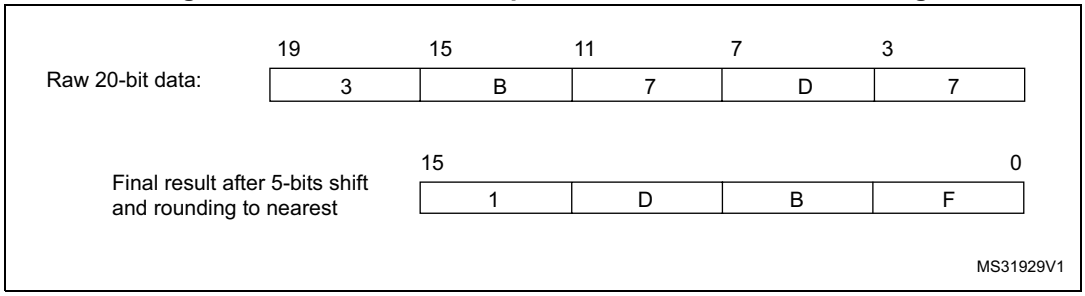
Note: If the intermediate result after the shifting exceeds 16 bits, the upper bits of the result are simply truncated.

Figure 105. 20-bit to 16-bit result truncation



The *Figure 106* gives a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

Figure 106. Numerical example with 5-bits shift and rounding



The *Table 173* below gives the data format for the various N and M combination, for a raw conversion data equal to 0xFFF.

Table 173. Maximum output results vs N and M. Grayed values indicates truncation

Oversampling ratio	Max Raw data	No-shift OVSS = 0000	1-bit shift OVSS = 0001	2-bit shift OVSS = 0010	3-bit shift OVSS = 0011	4-bit shift OVSS = 0100	5-bit shift OVSS = 0101	6-bit shift OVSS = 0110	7-bit shift OVSS = 0111	8-bit shift OVSS = 1000
2x	0x1FFE	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040	0x0020
4x	0x3FFC	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040
8x	0x7FF8	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080
16x	0xFFF0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100
32x	0x1FFE0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200
64x	0x3FFC0	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
128x	0x7FF80	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800
256x	0xFFF00	0xFF00	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF

The conversion timings in oversampler mode do not change compared to standard conversion mode: the sample time is maintained equal during the whole oversampling sequence. New data are provided every N conversion, with an equivalent delay equal to N x t_{CONV} = N x (t_{SMPL} + t_{SAR}). The flags features are raised as following:

- the end of the sampling phase (EOSMP) is set after each sampling phase
- the end of conversion (EOC) occurs once every N conversions, when the oversampled result is available
- the end of sequence (EOCSEQ) occurs once the sequence of oversampled data is completed (i.e. after N x sequence length conversions total)

ADC operating modes supported when oversampling

In oversampling mode, most of the ADC operating modes are available:

- Single or continuous mode conversions, forward or backward scanned sequences and up to 8 channels programmed sequence
- ADC conversions start either by software or with triggers
- ADC stop during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (WAIT, AUTOFF)
- Programmable resolution: in this case, the reduced conversion values (as per RES[1:0] bits in ADC_CFGR1 register) are accumulated, truncated, rounded and shifted in the same way as 12-bit conversions are

Note: The alignment mode is not available when working with oversampled data. The ALIGN bit in ADC_CFGR1 is ignored and the data are always provided right-aligned.

Analog watchdog

The analog watchdog functionality is available, with the following differences:

- the RES[1:0] bits are ignored, comparison is always done on using the full 12-bits values HTx[11:0] and LTx[11:0]
- the comparison is performed on the most significant 12 bits of the 16 bits oversampled results ADC_DR[15:4]

Note: Care must be taken when using high shifting values. This reduces the comparison range. For instance, if the oversampled result is shifted by 4 bits thus yielding a 12-bit data right-aligned, the effective analog watchdog comparison can only be performed on 8 bits. The comparison is done between ADC_DR[11:4] and HTx[7:0] / LTx[[7:0], and HTx[11:8] / LTx[11:8] must be kept reset.

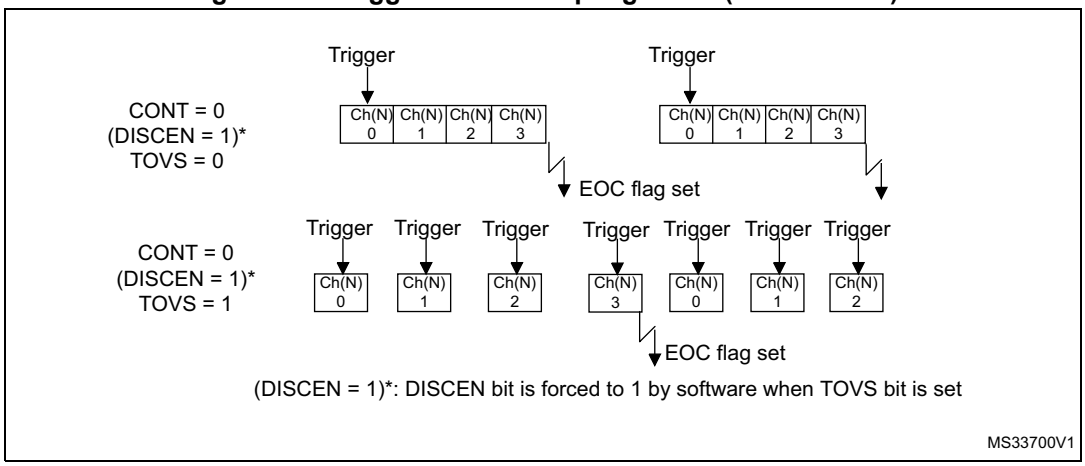
Triggered mode

The averager can also be used for basic filtering purposes. Although not a very efficient filter (slow roll-off and limited stop band attenuation), it can be used as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched mode power supply). For this purpose, a specific discontinuous mode can be enabled with TOVS bit in ADC_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

Figure 107 below shows how conversions are started in response to triggers in discontinuous mode.

If the TOVS bit is set, the content of the DISCEN bit is ignored and considered as 1.

Figure 107. Triggered oversampling mode (TOVS bit = 1)



22.4.27 Temperature sensor and internal reference voltage

The temperature sensor can be used to measure the junction temperature (T_J) of the device. The temperature sensor is internally connected an ADC internal input channel which is used to convert the sensor's output voltage to a digital value. The sampling time for the temperature sensor analog pin must be greater than the minimum T_{S_temp} value specified in the datasheet. When not in use, the sensor can be put in Power-down mode.

The internal voltage reference (V_{REFINT}) provides a stable (bandgap) voltage output for the ADC and the comparators.

Refer to Table *ADC interconnection* in *Section 22.4.2: ADC pins and internal signals* for details on the ADC internal input channel to which the above voltages are connected.

Figure 108 shows the block diagram of connections between the temperature sensor, the internal voltage reference and the ADC.

The VSENSESEL bit must be set to enable the conversion of V_{SENSE} while VREFEN bit must be set to enable the conversion of V_{REFINT}.

When the ADC operates in autonomous mode, these signals are controlled automatically to reduce power consumption (VSENSESEL and VREFEN must be set to measure the voltage in autonomous mode).

The temperature sensor output voltage linearly changes with the temperature. The offset of this line varies from chip to chip due to process variation (up to 45 °C from one chip to another).

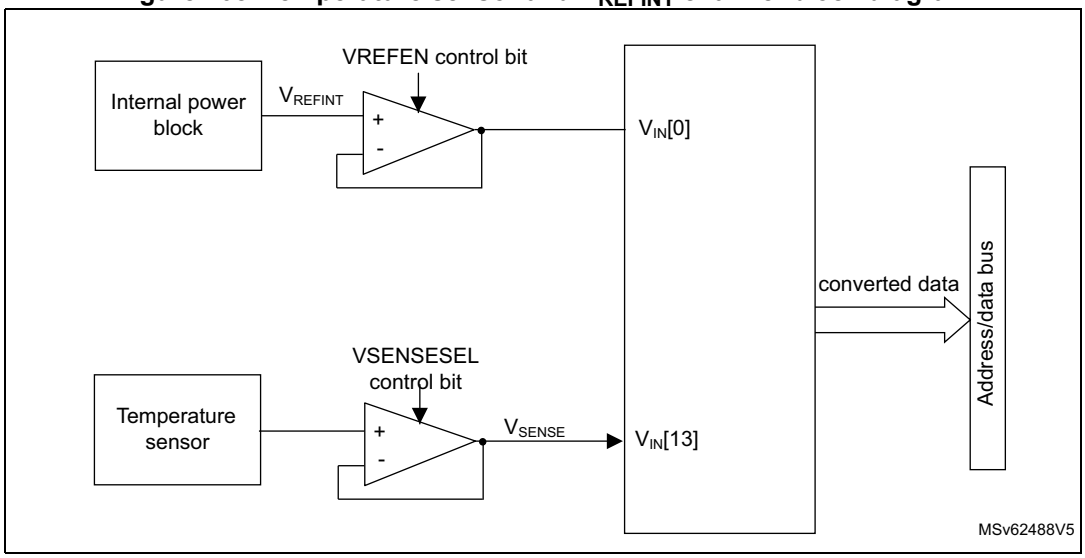
The uncalibrated internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. To improve the accuracy of the temperature sensor measurement, calibration values are stored in system memory for each device by STMicroelectronics during production.

During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference. Refer to the datasheet for additional information.

Main features

- Linearity: ±2 °C max., precision depending on calibration

Figure 108. Temperature sensor and V_{REFINT} channel block diagram



Reading the temperature

1. Select the input channel connected to V_{SENSE} (refer to Table *ADC interconnection* in *Section 22.4.2: ADC pins and internal signals*).
2. Select an appropriate sampling time specified in the device datasheet (T_{S_temp}).
3. Set the VSENSESEL bit in the ADC_CCR register to wake up the temperature sensor from power-down mode and wait for its stabilization time (t_{START}).
4. Start the ADC conversion by setting the ADSTART bit in the ADC_CR register (or by external trigger)
5. Read the resulting V_{SENSE} data in the ADC_DR register
6. Calculate the temperature using the following formula

$$\text{Temperature (in °C)} = \frac{\text{TS_CAL2_TEMP} - \text{TS_CAL1_TEMP}}{\text{TS_CAL2} - \text{TS_CAL1}} \times (\text{TS_DATA} - \text{TS_CAL1}) + \text{TS_CAL1_TEMP}$$

Where:

- TS_CAL2 is the temperature sensor calibration value acquired at TS_CAL2_TEMP.
- TS_CAL1 is the temperature sensor calibration value acquired at TS_CAL1_TEMP.
- TS_DATA is the actual temperature sensor output value converted by the ADC.

Refer to *Section 22.3: ADC implementation* for more information on TS_CAL1 and TS_CAL2 calibration points.

Note: The sensor has a startup time after waking up from power-down mode before it can output V_{SENSE} at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN and VSENSESEL bits must be set at the same time.

Calculating the actual V_{DDA} voltage using the internal reference voltage

The V_{DDA} power supply voltage applied to the microcontroller may be subject to variation or not precisely known. The embedded internal voltage reference (VREFINT) and its calibration data acquired by the ADC during the manufacturing process at V_{DDA} = 3.0 V can be used to evaluate the actual V_{DDA} voltage level.

The following formula gives the actual V_{DDA} voltage supplying the device:

$$V_{DDA} = 3.0\ V \times VREFINT_CAL / VREFINT_DATA$$

Where:

- VREFINT_CAL is the VREFINT calibration value
- VREFINT_DATA is the actual VREFINT output value converted by ADC

Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between the voltage reference V_{DDA} and the voltage applied on the converted channel. For most application use cases, it is necessary to convert this ratio into a voltage independent from V_{DDA}. For applications where V_{DDA} is known and ADC converted values are right-aligned, the following formula can be used to calculate this absolute value:

$$V_{CHANNELx} = \frac{V_{DDA}}{FULL_SCALE} \times ADC_DATA_x$$

For applications where V_{DDA} value is not known, the internal voltage reference and V_{DDA} can be replaced by the expression provided in *Section : Calculating the actual V_{DDA} voltage using the internal reference voltage*, resulting in the following formula:

$$V_{CHANNELx} = \frac{3.0\ V \times VREFINT_CAL \times ADC_DATA_x}{VREFINT_DATA \times FULL_SCALE}$$

Where:

- VREFINT_CAL is the VREFINT calibration value (refer to *Section 22.3: ADC implementation* for the value of VREFINT_CAL).
- ADC_DATA_x is the value measured by the ADC on channelx (right-aligned).
- VREFINT_DATA is the actual VREFINT output value converted by the ADC.
- FULL_SCALE is the maximum digital value of the ADC output. For example with 12-bit resolution, it is 2^{12} - 1 = 4095 or with 8-bit resolution, 2^8 - 1 = 255.

Note: If ADC measurements are done using an output format other than 12 bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

22.5 ADC in low-power modes

Table 174. Effect of low-power modes on the ADC

Mode	Description
Sleep	No effect, DMA requests are functional. ADC interrupts cause the device to exit Sleep mode.
Stop 0 and Stop 1	The content of the ADC register is kept. ADC can be functional. DMA request are functional, and the interrupt cause the device to exit Stop mode.
Stop 2, Stop 3, Standby	The ADC peripheral is powered down and must be reinitialized after exiting Stop 2, Stop 3, and Standby modes.

22.6 ADC interrupts

An interrupt can be generated by any of the following events:

- End of calibration (EOCAL flag)
- ADC power-up, when the ADC is ready (ADRDY flag)
- End of any conversion (EOC flag)
- End of a sequence of conversions (EOS flag)
- When an analog watchdog detection occurs (AWD1, AWD2, AWD3 flags)
- When the end of sampling phase occurs (EOSMP flag)
- when a data overrun occurs (OVR flag)
- LDO ready, when LDO output is stabilized (LDORDY flag)

Separate interrupt enable bits are available for flexibility.

Table 175. ADC wake-up and interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop mode(1)	Exit from Standby mode
ADC	LDO ready	LDORDY	LDORDYIE	Program LDORDY to 1	Yes	Yes	No
	End of calibration	EOCAL	EOCALIE	Program EOCAL to 1		Yes	
	ADC ready	ADRDY	ADRDYIE	Program ADRDY to 1		Yes	
	End of conversion	EOC	EOCIE	Program EOC to 1 OR read ADC_DR		Yes	
	End of sequence of conversions	EOS	EOSIE	Program EOS to 1		Yes	
	Analog watchdog 1 status bit is set	AWD1	AWD1IE	Program AWD1 to 1		Yes	
	Analog watchdog 2 status bit is set	AWD2	AWD2IE	Program AWD2 to 1		Yes	
	Analog watchdog 3 status bit is set	AWD3	AWD3IE	Program AWD2 to 1		Yes	
	End of sampling phase	EOSMP	EOSMPIE	Program EOSMP to 1		No	
	Overrun	OVR	OVRIE	Program OVR to 1		Yes	

1. The ADC can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to *Section 22.3: ADC implementation* for the list of supported Stop modes.

22.7 ADC registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

22.7.1 ADC interrupt and status register (ADC_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LDO RDY	EOCAL	Res.	AWD3	AWD2	AWD1	Res.	Res.	OVR	EOS	EOC	EOSMP	ADRDY
			rc_w1	rc_w1		rc_w1	rc_w1	rc_w1			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LDORDY**: LDO ready
This bit is set by hardware. It indicates that the ADC internal LDO output is ready.
It is cleared by software by writing 1 to it.
0: ADC voltage regulator disabled
1: ADC voltage regulator enabled and stabilized

Bit 11 **EOCAL**: End of calibration flag
This bit is set by hardware when calibration is complete. It is cleared by software writing 1 to it.
0: Calibration is not complete
1: Calibration is complete

Bit 10 Reserved, must be kept at reset value.

Bit 9 **AWD3**: Analog watchdog 3 flag
This bit is set by hardware when the converted voltage crosses the values programmed in ADC_AWD3TR and ADC_AWD3TR registers. It is cleared by software by writing 1 to it.
0: No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software)
1: Analog watchdog event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag
This bit is set by hardware when the converted voltage crosses the values programmed in ADC_AWD2TR and ADC_AWD2TR registers. It is cleared by software writing 1 to it.
0: No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software)
1: Analog watchdog event occurred

Bit 7 **AWD1**: Analog watchdog 1 flag
This bit is set by hardware when the converted voltage crosses the values programmed in ADC_TR1 and ADC_HR1 registers. It is cleared by software by writing 1 to it.
0: No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software)
1: Analog watchdog event occurred

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **OVR**: ADC overrun

This bit is set by hardware when an overrun occurs, meaning that a new conversion has complete while the EOC flag was already set. It is cleared by software writing 1 to it.

0: No overrun occurred (or the flag event was already acknowledged and cleared by software)

1: Overrun has occurred

Bit 3 **EOS**: End of sequence flag

This bit is set by hardware at the end of the conversion of a sequence of channels selected by the CHSEL bits. It is cleared by software writing 1 to it.

0: Conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Conversion sequence complete

Bit 2 **EOC**: End of conversion flag

This bit is set by hardware at the end of each conversion of a channel when a new data result is available in the ADC_DR register. It is cleared by software writing 1 to it or by reading the ADC_DR register.

0: Channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Channel conversion complete

Bit 1 **EOSMP**: End of sampling flag

This bit is set by hardware during the conversion, at the end of the sampling phase.It is cleared by software by writing 1 to it.

0: Not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)

1: End of sampling phase reached

Bit 0 **ADRDY**: ADC ready

This bit is set by hardware after the ADC has been enabled (ADEN = 1) and when the ADC reaches a state where it is ready to accept conversion requests.

It is cleared by software writing 1 to it.

0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)

1: ADC is ready to start conversion

22.7.2 ADC interrupt enable register (ADC_IER)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LDORDYIE	EOCALIE	Res.	AWD3IE	AWD2IE	AWD1IE	Res.	Res.	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
			rw	rw		rw	rw	rw			rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LDORDYIE**: LDO ready interrupt enable

This bit is set and cleared by software. It is used to enable/disable the LDORDY interrupt.

0: LDO ready interrupt disabled

1: LDO ready interrupt enabled. An interrupt is generated when the LDO output is ready.

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensure that no conversion is ongoing).

Bit 11 **EOCALIE**: End of calibration interrupt enable

This bit is set and cleared by software to enable/disable the end of calibration interrupt.

0: End of calibration interrupt disabled

1: End of calibration interrupt enabled

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog interrupt.

0: Analog watchdog interrupt disabled

1: Analog watchdog interrupt enabled

Note: The Software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 8 **AWD2IE**: Analog watchdog 2 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog interrupt.

0: Analog watchdog interrupt disabled

1: Analog watchdog interrupt enabled

Note: The Software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 7 **AWD1IE**: Analog watchdog 1 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog interrupt.

0: Analog watchdog interrupt disabled

1: Analog watchdog interrupt enabled

Note: The Software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **OVRIE**: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the overrun interrupt.

0: Overrun interrupt disabled

1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 3 **EOSIE**: End of conversion sequence interrupt enable

This bit is set and cleared by software to enable/disable the end of sequence of conversions interrupt.

0: EOS interrupt disabled

1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 2 **EOCIE**: End of conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of conversion interrupt.

0: EOC interrupt disabled

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 1 **EOSMPIE**: End of sampling flag interrupt enable

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt.

0: EOSMP interrupt disabled.

1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 0 **ADRDYIE**: ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.

0: ADRDY interrupt disabled.

1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

22.7.3 ADC control register (ADC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	Res.	Res.	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs			rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADSTP	Res.	ADSTART	ADDIS	ADEN
											rs		rs	rs	rs

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to start the calibration of the ADC.

It is cleared by hardware after calibration is complete.

0: Calibration complete

1: Write 1 to calibrate the ADC. Read at 1 means that a calibration is in progress.

Note: The software is allowed to set ADCAL only when the ADC is disabled (ADCAL = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0, AUTOFF = 0, and ADEN = 0).

The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN is set to 1 and ADSTART is cleared to 0 by writing ADSTP to 1 (ADC enabled and no conversion is ongoing).

Bits 30:29 Reserved, must be kept at reset value.

Bit 28 **ADVREGEN**: ADC voltage regulator enable

This bit is set by software, to enable the ADC internal voltage regulator. The voltage regulator output is available after $t_{ADCVREG_SETUP}$.

It is cleared by software to disable the voltage regulator. It can be cleared only if ADEN is et to 0.

0: ADC voltage regulator disabled

1: ADC voltage regulator enabled

Note: The software is allowed to program this bit field only when the ADC is disabled (ADCAL = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 27:5 Reserved, must be kept at reset value.

Bit 4 **ADSTP**: ADC stop conversion command

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 and the ADC is ready to accept a new start conversion command.

0: No ADC stop conversion command ongoing

1: Write 1 to stop the ADC. Read 1 means that an ADSTP command is in progress.

Note: To clear the A/D converter state, ADSTP must be set to 1 even if ADSTART is cleared to 0 after the software trigger A/D conversion. It is recommended to set ADSTP to 1 whenever the configuration needs to be modified.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **ADSTART**: ADC start conversion command

This bit is set by software to start ADC conversion. Depending on the EXTEN [1:0] configuration bits, a conversion either starts immediately (software trigger configuration) or once a hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– In single conversion mode (CONT = 0, DISCEN = 0), when software trigger is selected (EXTEN = 00): at the assertion of the end of Conversion Sequence (EOS) flag.

– In discontinuous conversion mode(CONT=0, DISCEN = 1), when the software trigger is selected (EXTEN = 00): at the assertion of the end of Conversion (EOC) flag.

– In all other cases: after the execution of the ADSTP command, at the same time as the ADSTP bit is cleared by hardware.

0: No ADC conversion is ongoing.

1: Write 1 to start the ADC. Read 1 means that the ADC is operating and may be converting.

Note: The software is allowed to set ADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC).

Bit 1 **ADDIS**: ADC disable command

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).

0: No ADDIS command ongoing

1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

Note: Setting ADDIS to 1 is only effective when ADEN = 1 and ADSTART = 0 (which ensures that no conversion is ongoing)

Bit 0 **ADEN**: ADC enable command

This bit is set by software to enable the ADC. The ADC is 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.

0: ADC is disabled (OFF state)

1: Write 1 to enable the ADC.

Note: The software is allowed to set ADEN only when all bits of ADC_CR registers are 0 (ADCAL = 0, ADSTP = 0, ADSTART = 0, ADDIS = 0 and ADEN = 0)

22.7.4 ADC configuration register 1 (ADC_CFGR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AWD1CH[4:0]					Res.	Res.	AWD1 EN	AWD1 SGL	CHSEL RMOD	Res.	Res.	Res.	Res.	DISCEN
	rw	rw	rw	rw	rw			rw	rw	rw					rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	WAIT	CONT	OVRMOD	EXTEN[1:0]		Res.	EXTSEL[2:0]			ALIGN	SCAN DIR	RES[1:0]		DMAC FG	DMAEN
	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: ADC analog input Channel 0 monitored by AWD

00001: ADC analog input Channel 1 monitored by AWD

.....

01101: ADC analog input Channel 13 monitored by AWD

Others: Reserved

Note: The channel selected by the AWDCH[4:0] bits must be also set into the CHSELR register. The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bits 25:24 Reserved, must be kept at reset value.

Bit 23 **AWD1EN**: Analog watchdog enable

This bit is set and cleared by software.

0: Analog watchdog 1 disabled

1: Analog watchdog 1 enabled

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 22 **AWD1SGL**: Enable the watchdog on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWDCH[4:0] bits or on all the channels

0: Analog watchdog 1 enabled on all channels

1: Analog watchdog 1 enabled on a single channel

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 21 **CHSELRMOD**: Mode selection of the ADC_CHSELR register

This bit is set and cleared by software to control the ADC_CHSELR feature:

0: Each bit of the ADC_CHSELR register enables an input

1: ADC_CHSELR register is able to sequence up to 8 channels

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bits 20:17 Reserved, must be kept at reset value.

Bit 16 **DISCEN**: Discontinuous mode

This bit is set and cleared by software to enable/disable discontinuous mode.

0: Discontinuous mode disabled

1: Discontinuous mode enabled

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.

The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **WAIT**: Wait conversion mode

This bit is set and cleared by software to enable/disable wait conversion mode.

0: Wait conversion mode off

1: Wait conversion mode on

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 13 **CONT**: Single / continuous conversion mode

This bit is set and cleared by software. If it is set, conversion takes place continuously until it is cleared.

0: Single conversion mode

1: Continuous conversion mode

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.

The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 12 **OVRMOD**: Overrun management mode

This bit is set and cleared by software and configure the way data overruns are managed.

0: ADC_DR register is preserved with the old data when an overrun is detected.

1: ADC_DR register is overwritten with the last conversion result when an overrun is detected.

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection

These bits are set and cleared by software to select the external trigger polarity and enable the trigger.

00: Hardware trigger detection disabled (conversions can be started by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 9 Reserved, must be kept at reset value.

Bits 8:6 **EXTSEL[2:0]**: External trigger selection

These bits select the external event used to trigger the start of conversion (refer to table *ADC interconnection* in *Section 22.4.2: ADC pins and internal signals* for details):

000: adc_trg0
001: adc_trg1
010: adc_trg2
011: adc_trg3
100: adc_trg4
101: adc_trg5
110: adc_trg6
111: adc_trg7

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 5 **ALIGN**: Data alignment

This bit is set and cleared by software to select right or left alignment. Refer to *Figure 93: Data alignment and resolution (oversampling disabled: OVSE = 0)*

0: Right alignment
1: Left alignment

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 4 **SCANDIR**: Scan sequence direction

This bit is set and cleared by software to select the direction in which the channels is scanned in the sequence. It is effective only if CHSELRMOD bit is cleared to 0.

0: Upward scan (from CHSEL0 to CHSEL13)
1: Backward scan (from CHSEL13 to CHSEL0)

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bits 3:2 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

00: 12 bits
01: 10 bits
10: 8 bits
11: 6 bits

Note: The software is allowed to write these bits only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 1 **DMACFG**: Direct memory access configuration

This bit is set and cleared by software to select between two DMA modes of operation and is effective only when DMAEN = 1.

0: DMA one-shot mode selected

1: DMA circular mode selected

For more details, refer to *Managing converted data using the DMA*.

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 0 **DMAEN**: Direct memory access enable

This bit is set and cleared by software to enable the generation of DMA requests. This allows the automatic management of the converted data by the DMA controller. For more details, refer to *Managing converted data using the DMA*.

0: DMA disabled

1: DMA enabled

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

22.7.5 ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	LFTRIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rw													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TOVS	OVSS[3:0]				OVSR[2:0]			Res.	OVSE
						rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **LFTRIG**: Low-frequency trigger mode enable

This bit must be set by software.

0: Reserved

1: Low-frequency trigger mode enabled.

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bits 28:10 Reserved, must be kept at reset value.

Bit 9 **TOVS**: Triggered *Oversampling*

This bit is set and cleared by software.

0: All oversampled conversions for a channel are done consecutively after a trigger

1: Each oversampled conversion for a channel needs a trigger

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bits 8:5 **OVSS[3:0]**: *Oversampling shift*

This bit is set and cleared by software.

0000: No shift

0001: Shift 1-bit

0010: Shift 2-bits

0011: Shift 3-bits

0100: Shift 4-bits

0101: Shift 5-bits

0110: Shift 6-bits

0111: Shift 7-bits

1000: Shift 8-bits

Others: Reserved

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1(which ensures that no conversion is ongoing).

Bits 4:2 **OVSR[2:0]**: *Oversampling ratio*

This bit filed defines the number of oversampling ratio.

000: 2x

001: 4x

010: 8x

011: 16x

100: 32x

101: 64x

110: 128x

111: 256x

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bit 1 Reserved, must be kept at reset value.

Bit 0 **OVSE**: *Oversampler Enable*

This bit is set and cleared by software.

0: Oversampler disabled

1: Oversampler enabled

Note: Software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

22.7.6 ADC sampling time register (ADC_SMPR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPSEL13	SMPSEL12	SMPSEL11	SMPSEL10	SMPSEL9	SMPSEL8
										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMPSEL7	SMPSEL6	SMPSEL5	SMPSEL4	SMPSEL3	SMPSEL2	SMPSEL1	SMPSEL0	Res.	SMP2[2:0]			Res.	SMP1[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:8 **SMPSELx:** Channel-x sampling time selection (x = 13 to 0)

These bits are written by software to define which sampling time is used.

0: Sampling time of CHANNELx use the setting of SMP1[2:0] register.

1: Sampling time of CHANNELx use the setting of SMP2[2:0] register.

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **SMP2[2:0]:** Sampling time selection 2

These bits are written by software to select the sampling time that applies to all channels.

000: 1.5 ADC clock cycles

001: 3.5 ADC clock cycles

010: 7.5 ADC clock cycles

011: 12.5 ADC clock cycles

100: 19.5 ADC clock cycles

101: 39.5 ADC clock cycles

110: 79.5 ADC clock cycles

111: 814.5 ADC clock cycles

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SMP1[2:0]:** *Sampling time selection 1*

These bits are written by software to select the sampling time that applies to all channels.

000: 1.5 ADC clock cycles

001: 3.5 ADC clock cycles

010: 7.5 ADC clock cycles

011: 12.5 ADC clock cycles

100: 19.5 ADC clock cycles

101: 39.5 ADC clock cycles

110: 79.5 ADC clock cycles

111: 814.5 ADC clock cycles

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

22.7.7 ADC watchdog threshold register (ADC_AWD1TR)

Address offset: 0x20

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HT1[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LT1[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HT1[11:0]**: Analog watchdog 1 higher threshold
These bits are written by software to define the higher threshold for the analog watchdog.
Refer to *Section 22.4.25: Analog window watchdog*.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **LT1[11:0]**: Analog watchdog *1* lower threshold
These bits are written by software to define the lower threshold for the analog watchdog.
Refer to *Section 22.4.25: Analog window watchdog*.

22.7.8 ADC watchdog threshold register (ADC_AWD2TR)

Address offset: 0x24

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HT2[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LT2[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HT2[11:0]**: Analog watchdog 2 higher threshold
These bits are written by software to define the higher threshold for the analog watchdog.
Refer to *Section 22.4.25: Analog window watchdog*.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **LT2[11:0]**: Analog watchdog 2 lower threshold
These bits are written by software to define the lower threshold for the analog watchdog.
Refer to *Section 22.4.25: Analog window watchdog*.

22.7.9 ADC channel selection register [alternate] (ADC_CHSELR)

Address offset: 0x28

Reset value: 0x0000 0000

The same register can be used in two different modes:

– Each ADC_CHSELR bit enables an input (CHSELRMOD = 0 in ADC_CFGR1). Refer to the current section.

– ADC_CHSELR is able to sequence up to 8 channels (CHSELRMOD = 1 in ADC_CFGR1). Refer to next section.

CHSELRMOD = 0 in ADC_CFGR1

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

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **CHSELx**: Channel x selection (x = 13 to 0)

These bits are written by software and define which channels are part of the sequence of channels to be converted.

0: Input Channel-x is not selected for conversion

1: Input Channel-x is selected for conversion

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

22.7.10 ADC channel selection register [alternate] (ADC_CHSELR)

Address offset: 0x28

Reset value: 0x0000 0000

The same register can be used in two different modes:

– Each ADC_CHSELR bit enables an input (CHSELRMOD = 0 in ADC_CFGR1). Refer to the current previous section.

– ADC_CHSELR is able to sequence up to 8 channels (CHSELRMOD = 1 in ADC_CFGR1). Refer to this section.

CHSELRMOD = 1 in ADC_CFGR1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SQ8[3:0]				SQ7[3:0]				SQ6[3:0]				SQ5[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4[3:0]				SQ3[3:0]				SQ2[3:0]				SQ1[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **SQ8[3:0]**: 8th conversion of the sequence

These bits are programmed by software with the channel number assigned to the 8th conversion of the sequence. 0b1111 indicates the end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

0000: CH0

0001: CH1

...

1101: CH13

1110: Reserved

1111: No channel selected (End of sequence)

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bits 27:24 **SQ7[3:0]**: 7th conversion of the sequence

These bits are programmed by software with the channel number assigned to the 7th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bits 23:20 **SQ6[3:0]**: 6th conversion of the sequence

These bits are programmed by software with the channel number assigned to the 6th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bits 19:16 **SQ5[3:0]**: 5th conversion of the sequence

These bits are programmed by software with the channel number assigned to the 5th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bits 15:12 **SQ4[3:0]**: 4th conversion of the sequence

These bits are programmed by software with the channel number assigned to the 4th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1(which ensures that no conversion is ongoing).

Bits 11:8 **SQ3[3:0]**: 3rd conversion of the sequence

These bits are programmed by software with the channel number assigned to the 3rd conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bits 7:4 **SQ2[3:0]**: 2nd conversion of the sequence

These bits are programmed by software with the channel number assigned to the 2nd conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bits 3:0 **SQ1[3:0]**: 1st conversion of the sequence

These bits are programmed by software with the channel number assigned to the 1st conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

22.7.11 ADC watchdog threshold register (ADC_AWD3TR)

Address offset: 0x2C

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HT3[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LT3[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HT3[11:0]**: *Analog watchdog 3 higher threshold*

These bits are written by software to define the higher threshold for the analog watchdog.

Refer to *Section 22.4.25: Analog window watchdog*.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **LT3[11:0]**: Analog watchdog *3*lower threshold

These bits are written by software to define the lower threshold for the analog watchdog.

Refer to *Section 22.4.25: Analog window watchdog*.

22.7.12 ADC data register (ADC_DR)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DATA[15:0]**: Converted data

These bits are read-only. They contain the conversion result from the last converted channel. The data are left- or right-aligned as shown in *Figure 93: Data alignment and resolution (oversampling disabled: OVSE = 0)*.

Just after a calibration is complete, DATA[6:0] contains the calibration factor.

22.7.13 ADC power register (ADC_PWRR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DPD	AUTOFF
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DPD**: Deep-power-down mode bit

This bit is set and cleared by software. It is used to enable/disable Deep-power-down mode in autonomous mode when the ADC is not used.

0: Deep-power-down mode disabled

1: Deep-power-down mode enabled

Note: The software is allowed to write this bit only when ADEN bit is cleared to 0 (this ensures that no conversion is ongoing).

Setting DPD in auto-off mode automatically disables the LDO.

Bit 0 **AUTOFF**: Auto-off mode bit

This bit is set and cleared by software. it is used to enable/disable the auto-off mode.

0: Auto-off mode disabled

1: Auto-off mode enabled

Note: The software is allowed to write this bit only when ADEN bit is cleared to 0 (this ensures that no conversion is ongoing).

22.7.14 ADC Analog Watchdog 2 Configuration register (ADC_AWD2CR)

Address offset: 0xA0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	AWD2 CH13	AWD2 CH12	AWD2 CH11	AWD2 CH10	AWD2 CH9	AWD2 CH8	AWD2 CH7	AWD2 CH6	AWD2 CH5	AWD2 CH4	AWD2 CH3	AWD2 CH2	AWD2 CH1	AWD2 CH0
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **AWD2CHx:** Analog watchdog channel selection (x = 13 to 0)

These bits are set and cleared by software. They enable and select the input channels to be guarded by analog watchdog 2 (AWD2).

0: ADC analog channel-x is not monitored by AWD2

1: ADC analog channel-x is monitored by AWD2

Note: The channels selected through ADC_AWD2CR must be also configured into the ADC_CHSELR registers. Refer to SQ8[3:0] for a definition of channel selection. The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

22.7.15 ADC Analog Watchdog 3 Configuration register (ADC_AWD3CR)

Address offset: 0xA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	AWD3 CH13	AWD3 CH12	AWD3 CH11	AWD3 CH10	AWD3 CH9	AWD3 CH8	AWD3 CH7	AWD3 CH6	AWD3 CH5	AWD3 CH4	AWD3 CH3	AWD3 CH2	AWD3 CH1	AWD3 CH0
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **AWD3CHx:** Analog watchdog channel selection (x = 13 to 0)

These bits are set and cleared by software. They enable and select the input channels to be guarded by analog watchdog 3 (AWD3).

0: ADC analog channel-x is not monitored by AWD3

1: ADC analog channel-x is monitored by AWD3

Note: The channels selected through ADC_AWD3CR must be also configured into the ADC_CHSELR registers. Refer to SQ8[3:0] for a definition of channel selection. The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

22.7.16 ADC Calibration factor (ADC_CALFACT)

Address offset: 0xC4

Reset value: 0x0000 0000

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

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **CALFACT[6:0]**: Calibration factor

These bits are written by hardware or by software.

– Once a calibration is complete, they are updated by hardware with the calibration factors.

– Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is then applied once a new calibration is launched.

– Just after a calibration is complete, DATA[6:0] contains the calibration factor.

Note: Software can write these bits only when ADEN = 1 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

22.7.17 ADC common configuration register (ADC_CCR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VSENSE SEL	VREF EN	PRESC[3:0]				Res.	Res.
								rw	rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **VSENSESEL**: Temperature sensor selection

This bit is set and cleared by software to enable/disable the temperature sensor.

0: Temperature sensor disabled

1: Temperature sensor enabled

Note: Software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bit 22 **VREFEN**: V_{REFINT} enable

This bit is set and cleared by software to enable/disable the V_{REFINT} buffer.

0: V_{REFINT} disabled

1: V_{REFINT} enabled

Note: Software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bits 21:18 **PRESC[3:0]:** ADC prescaler

Set and cleared by software to select the frequency of the clock to the ADC.

0000: input ADC clock not divided
0001: input ADC clock divided by 2
0010: input ADC clock divided by 4
0011: input ADC clock divided by 6
0100: input ADC clock divided by 8
0101: input ADC clock divided by 10
0110: input ADC clock divided by 12
0111: input ADC clock divided by 16
1000: input ADC clock divided by 32
1001: input ADC clock divided by 64
1010: input ADC clock divided by 128
1011: input ADC clock divided by 256
Other: Reserved

Note: Software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 17:0 Reserved, must be kept at reset value.

22.7.18 ADC register map

The following table summarizes the ADC registers.

Table 176. ADC register map and reset values

Offset	Register name	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
0x00	ADC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDORDY	EOCAL	Res.	AWD3	AWD2	AWD1	Res.	Res.	OVR	EOS	EOC	EOSMP	ADRDY
	Reset value																				0	0		0	0	0			0	0	0	0	0
0x04	ADC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDORDYIE	EOCALIE	Res.	AWD3IE	AWD2IE	AWD1IE	Res.	Res.	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																				0	0		0	0	0			0	0	0	0	0
0x08	ADC_CR	ADCAL	Res.	Res.	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADSTP	Res.	ADSTART	ADDIS	ADEN
	Reset value	0			0																								0		0	0	0
0x0C	ADC_CFGR1	Res.	AWDCH[4:0]					Res.	Res.	AWD1EN	AWD1SG	CHSELRMOD	Res.	Res.	Res.	Res.	DISCEN	Res.	WAIT	CONT	OVRMOD	EXTEN[1:0]		Res.	EXTSEL[2:0]			ALIGN	SCANDIR	RES[1:0]		DMACFG	DMAEN
	Reset value		0	0	0	0	0			0	0	0					0		0	0	0	0	0		0	0	0	0	0	0	0	0	0

Table 176. ADC register map and reset values (continued)

Offset	Register name	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
0x10	ADC_CFGR2	Res.	Res.	LFTRIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOVS	OVSS[3:0]				OVSR [2:0]			Res.	OVSE
	Reset value			0																				0	0	0	0	0	0	0	0		0
0x14	ADC_SMPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPSEL13	SMPSEL12	SMPSEL11	SMPSEL10	SMPSEL9	SMPSEL8	SMPSEL7	SMPSEL6	SMPSEL5	SMPSEL4	SMPSEL3	SMPSEL2	SMPSEL1	SMPSEL0	Res.	SMP2 [2:0]			Res.	SMP1 [2:0]		
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0		0	0	0
0x18-0x1C	Reserved	Reserved																															
0x20	ADC_AWD1TR	Res.	Res.	Res.	Res.	HT1[11:0]												Res.	Res.	Res.	Res.	LT1[11:0]											
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x24	ADC_AWD2TR	Res.	Res.	Res.	Res.	HT2[11:0]												Res.	Res.	Res.	Res.	LT2[11:0]											
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x28	ADC_CHSELR (CHSELRMOD = 0)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHSEL13	CHSEL12	CHSEL11	CHSEL10	CHSEL9	CHSEL8	CHSEL7	CHSEL6	CHSEL5	CHSEL4	CHSEL3	CHSEL2	CHSEL1	CHSEL0
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	ADC_CHSELR (CHSELRMOD = 1)	SQ8[3:0]				SQ7[3:0]				SQ6[3:0]				SQ5[3:0]				SQ4[3:0]				SQ3[3:0]				SQ2[3:0]				SQ1[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	ADC_AWD3TR	Res.	Res.	Res.	Res.	HT3[11:0]												Res.	Res.	Res.	Res.	LT3[11:0]											
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x30 - 0x3C	Reserved	Reserved																															
0x40	ADC_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	ADC_PWRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DPD	AUTOFF
	Reset value																															0	0
0x48 - 0xBF	Reserved	Reserved																															
0xA0	ADC_AWD2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD2CH13	AWD2CH12	AWD2CH11	AWD2CH10	AWD2CH9	AWD2CH8	AWD2CH7	AWD2CH6	AWD2CH5	AWD2CH4	AWD2CH3	AWD2CH2	AWD2CH1	AWD2CH0
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4	ADC_AWD3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3CH13	AWD3CH12	AWD3CH11	AWD3CH10	AWD3CH9	AWD3CH8	AWD3CH7	AWD3CH6	AWD3CH5	AWD3CH4	AWD3CH3	AWD3CH2	AWD3CH1	AWD3CH0
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4 - 0xC0	Reserved	Reserved																															
0xC4	ADC_CALFACT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT[6:0]						
	Reset value																										0	0	0	0	0	0	0
0xB8 - 0xCF	Reserved	Reserved																															
0x308	ADC_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VSENSESEL	VREFEN	PRESC3	PRESC2	PRESC1	PRESC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0	0	0	0	0																		

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

23 True random number generator (RNG)

23.1 RNG introduction

The RNG is a true random number generator that provides full entropy outputs to the application as 32-bit samples. It is composed of a live entropy source (analog) and an internal conditioning component.

The RNG is an NIST SP 800-90B compliant entropy source that can be used to construct a nondeterministic random bit generator (NDRBG).

The RNG true random number generator can be certified NIST SP 800-90B. It can also be tested using the German BSI statistical tests of AIS-31 (T0 to T8).

23.2 RNG main features

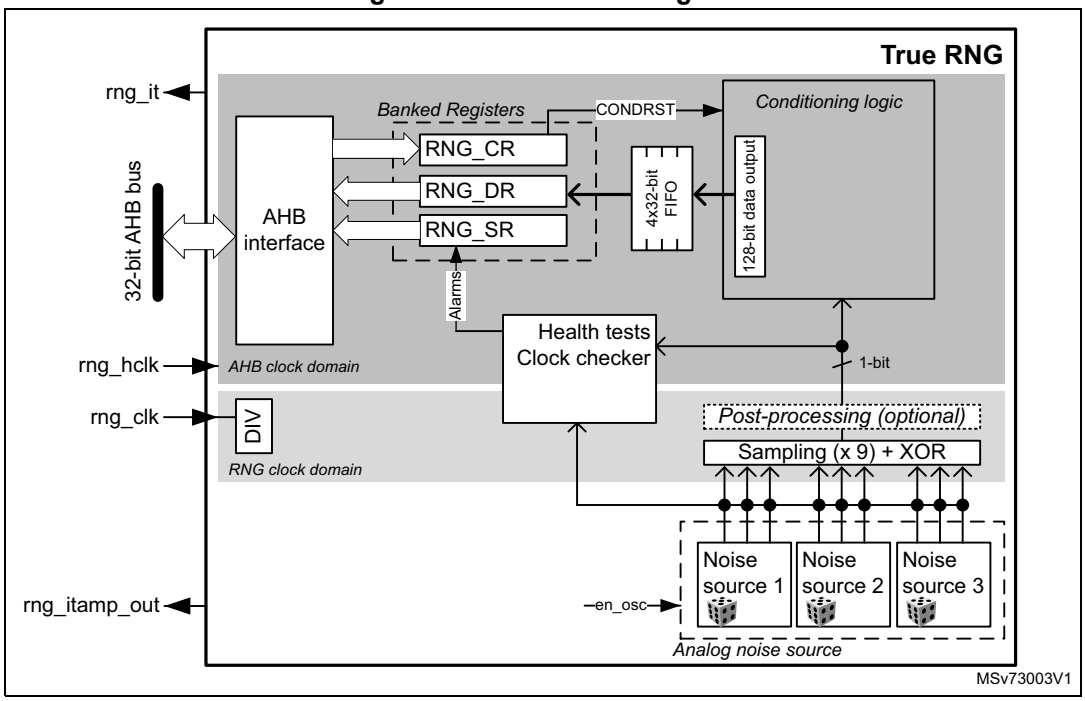
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source conditioned by an NIST SP800-90B approved conditioning stage.
- It can be used as the entropy source to construct a non-deterministic random bit generator (NDRBG).
- In the default configuration, it produces four 32-bit random samples every 412 AHB clock cycles if $f_{AHB} < f_{threshold}$ (256 RNG clock cycles otherwise).
- Embeds startup and NIST SP800-90B approved continuous health tests (repetition count and adaptive proportion tests), associated with specific error management.
- Can be disabled to reduce power consumption, or enabled with an automatic low power mode (default configuration).
- Has an AMBA® AHB slave peripheral, accessible through 32-bit word single accesses only (else an AHB bus error is generated, and the write accesses are ignored).

23.3 RNG functional description

23.3.1 RNG block diagram

Figure 109 shows the RNG block diagram.

Figure 109. RNG block diagram



23.3.2 RNG internal signals

Table 177 describes internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 177. RNG internal input/output signals

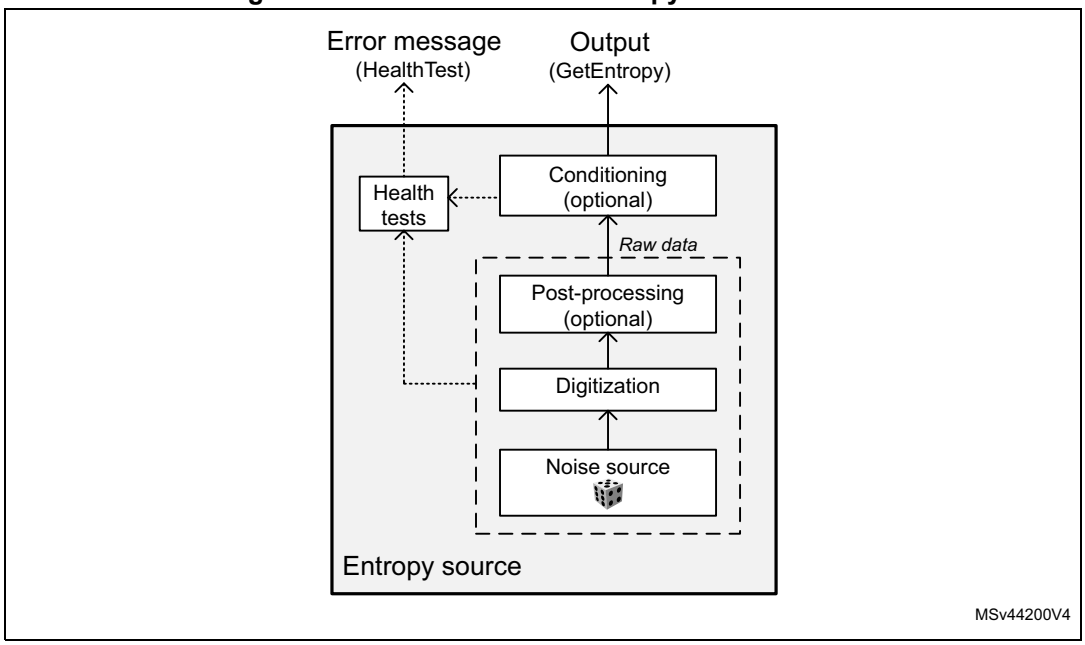
Signal name	Signal type	Description
rng_it	Digital output	RNG global interrupt request
rng_hclk	Digital input	AHB clock
rng_clk	Digital input	RNG dedicated clock, asynchronous to rng_hclk
rng_itamp_out	Digital output	RNG internal tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, RNG stops delivering random samples. A reset and a new initialization are needed to use it again.

23.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals.

Within its boundary, it integrates all the required NIST components shown in *Figure 110*: an analog noise source, a digitization stage, a conditioning algorithm, a health monitoring block, and two interfaces used to interact with the entropy source (GetEntropy and HealthTest).

Figure 110. NIST SP800-90B entropy source model



Noise source

This component contains the nondeterministic, entropy-providing activity, ultimately responsible for the uncertainty associated with the bit string output by the entropy source. This noise source provides 1-bit samples. It is composed of:

- Multiple analog noise sources (x3), each based on three XOR-ed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in *Section 23.3.8*. Multiple oscillators are also disabled for configuration A (see *Table 179*).
- The XOR-ing of all the noise sources into a single analog output.
- A sampling stage of this output is clocked by a dedicated clock input (rng_clk with integrated divider), delivering a 1-bit raw data output.

This noise source sampling is independent from the AHB interface clock frequency (rng_hclk), with the possibility, for the software, to decrease the sampling frequency by using the integrated divider.

Note: The recommended clock frequencies and associated divider value are detailed in Section 23.6.

Post processing

In the NIST configuration no post-processing is applied to the sampled noise source. In the non-NIST configuration B (as defined in *Section 23.6.2*) a normalization debiasing is applied: half of the bits are taken from the sampled noise source, the other half from the inverted sampled noise source.

Conditioning

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bit string output (128-bit). The NIST SP800-90B target is full entropy on the output (128-bit).

The time intervals required between two random number generations, and between the RNG initialization and availability of the first sample are detailed in *Section 23.5*.

Output buffer

A data output buffer can store up to four 32-bit words that have been output from the conditioning component. When four words have been read from the output FIFO through the RNG_DR register, the content of the 128-bit conditioning output register is pushed into the output FIFO, and a new conditioning round is automatically started. Four new words are added to the conditioning output register after a number of clock cycles specified in *Section 23.5*.

Whenever a random number is available through the RNG_DR register, the DRDY flag changes from 0 to 1. This flag remains high until the output buffer becomes empty after reading four words from the RNG_DR register.

Note: When the interrupts are enabled, an interrupt is generated when this flag transitions from 0 to 1. The interrupt is then cleared automatically by the RNG, as explained above.

Health checks

This component ensures that the entire entropy source (with its noise source) starts, then operates as expected, obtaining the assurance that failures are caught quickly, with high probability and reliability.

The RNG implements the following health check features in accordance with NIST SP800-90B. The described thresholds correspond to the default RNG_HTCR0 configuration.

1. Startup health tests, performed after reset and before the first use of the RNG as entropy source:
   - Repetition count test, flagging an error when the noise source has provided more than 28 consecutive bits at a constant value (0 or 1).
   - Adaptive proportion test running on a window of 1024 consecutive bits: the RNG verifies that the first bit on the outputs of the noise source is not repeated more than 684 times.
   - Known-answer tests, to verify the conditioning stage.
2. Continuous health tests, running indefinitely on the outputs of the noise source (XORed output and individual oscillator outputs, see *Figure 109: RNG block diagram*):
   - Repetition count test, similar to the one running in startup tests.
   - Adaptive proportion test, similar to the one running in startup tests.
3. Vendor specific continuous test:

- Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle (before divider) is smaller than the AHB clock cycle divided by 32.

4. On-demand test of digitized noise source (raw data)
   - Supported by restarting the entropy source and rerunning the startup tests (see software reset sequence in *Section 23.3.4*). Other kinds of on-demand testing (software based) are *not supported*.

The CECS and SECS status bits in the RNG_SR register indicate when an error condition is detected, as detailed in *Section 23.3.7*.

Note: An interrupt can be generated when an error is detected.

The health test thresholds are modified by changing the value in the corresponding RNG_HTCRx registers. See Section 23.6 for details.

23.3.4 RNG initialization

The RNG simplified state machine is shown in *Figure 111*.

After enabling the RNG (RNGEN = 1 in RNG_CR), the following chain of events occurs:

1. The analog noise source is enabled, and by default the RNG waits 16 RNG clock cycles (before divider) before starting to sample the analog output and filling the 128-bit conditioning shift register.
2. The conditioning hardware initializes, automatically triggering the startup behavior test on the raw data samples and known-answer tests.
3. Start-up health tests are completed, with the conditioning stage ready to use.
4. The conditioning stage internal input data buffer is filled again with 128-bit and a number of conditioning rounds defined by the RNG configuration (NIST or non-NIST) is performed. The output buffer is then filled with the post processing result.
5. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in *Section 23.5*.

Figure 111. RNG initialization overview

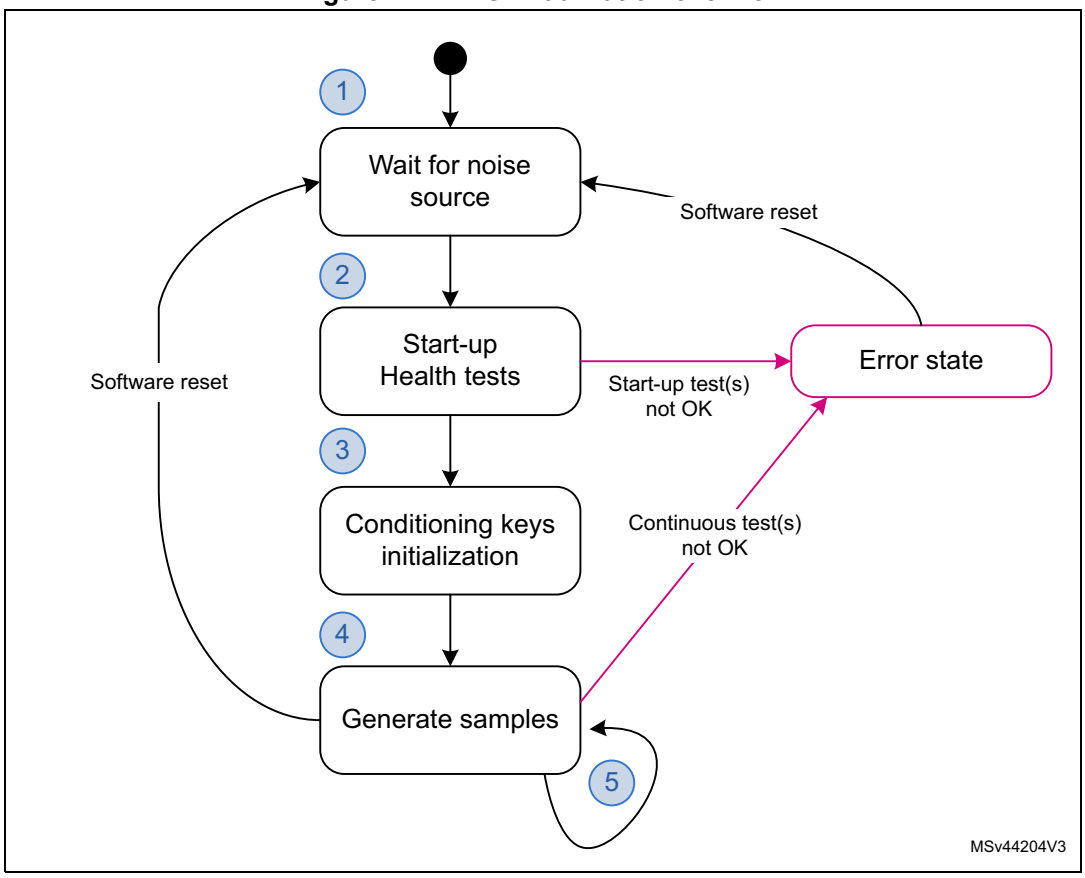


Figure 111 also highlights a possible software reset sequence, implemented by:

1. Write RNGEN = 0, then wait for BUSY = 0 in the RNG_SR register.
2. Write bits RNGEN = 0 and CONDRST = 1 in the RNG_CR register with the same RNG configuration and a new CLKDIV if needed.
3. Write RNGEN = 1 and CONDRST = 0 in the RNG_CR register.
4. Wait for a random number to be ready, after initialization completes.

Note: When the RNG is reset by RCC (hardware reset), the configuration for optimal randomness is lost in the registers.

23.3.5 RNG operation

Normal operation

To run the RNG using interrupts, the following steps are recommended:

1. Check in *Section 23.6* if a specific RNG configuration is required for the application.
   – If this is the case, when bits RNGEN and BUSY=0, write in the RNG_CR register bit CONDRST = 1, together with the correct RNG configuration. Then perform a second write with bit CONDRST = 0, interrupt enable bit IE = 1, and RNG enable bit RNGEN = 1.

- If this is not the case, perform a write to the RNG_CR register with interrupt enable bit IE = 1 and RNG enable bit RNGEN = 1.

2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore, at each interrupt, check that:
   - No error occurred. The SEIS and CEIS bits must be set to 0 in the RNG_SR register.
   - A random number is ready. The DRDY bit must be set to 1 in the RNG_SR register.
   - If the above two conditions are true the content of the RNG_DR register can be read up to four consecutive times. If valid data are available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in *Section 23.3.7* must be used.

To run the RNG in polling mode the following steps are recommended:

1. Check in *Section 23.6* if a specific RNG configuration is required for the application.
   - If this is the case, when bits RNGEN and BUSY = 0, write in the RNG_CR register bit CONDRST = 1, together with the correction RNG configuration. Then perform a second write to the RNG_CR register with bit CONDRST = 0 and the RNG enable bit RNGEN = 1.
   - If this is not the case, only enable the RNG by setting the RNGEN bit to 1 in the RNG_CR register.
2. Read the RNG_SR register and check that:
   - No error occurred (the SEIS and CEIS bits must be set to 0)
   - A random number is ready (the DRDY bit must be set to 1)
3. If the above conditions are true, read the content of the RNG_DR register up to four consecutive times. If valid data are available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, use the recovery sequence described in *Section 23.3.7*.

Note: When data are not ready (DRDY = 0) RNG_DR returns 0. It is recommended to verify that RNG_DR is different from 0. When this happens, a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

If the random number generation period is a concern to the application and if NIST compliance is not required, it is possible to select a faster configuration by using the configuration B, described in *Section 23.6*. The gain in random number generation speed is summarized in *Section 23.5*.

Low-power operation

If power consumption is a concern, low-power strategies can be used, as described in *Section 23.3.8*.

Software post-processing

No specific software post-processing/conditioning is expected to meet the AIS-31 or NIST SP800-90B approvals.

Built-in health check functions are described in *Section 23.3.3*.

23.3.6 RNG clocking

The RNG runs on two different clocks: the AHB bus clock and a dedicated RNG clock.

The AHB clock is used to clock the AHB banked registers and conditioning component. The RNG clock, coupled with a programmable divider (see CLKDIV bitfield in the RNG_CR register) is used for noise source sampling. Recommended clock configurations are detailed in *Section 23.6*.

*Note: When the CED bit in the RNG_CR register is set to 0, the RNG clock frequency before the internal divider **must be higher** than the AHB clock frequency divided by 32, otherwise the clock checker flags a clock error (CECS = 1 in the RNG_SR register).*

See *Section 23.3.1* for details (AHB and RNG clock domains).

23.3.7 Error management

In parallel to random number generation a health check block verifies the correct noise source behavior and the frequency of the RNG source clock as detailed in this section. Associated error state is also described.

Clock error detection

When the clock error detection is enabled (CED = 0) and if the RNG clock frequency is too low, the RNG sets to 1 both the CEIS and CECS bits to indicate that a clock error occurred. In this case, the application must check that the RNG clock is configured correctly (see *Section 23.3.6*), and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when the clocking condition is normal.

When CECS is set, values are available in the RNG_DR register, but it must not be used as described in *Section 23.3.5*.

Note: CEIS is set only when CECS is set to 1 by RNG.

Noise source error detection

When a noise source (or seed) error occurs, the RNG stops generating random numbers and sets to 1 both SEIS and SECS bits to indicate that a seed error occurred.

When SECS is set, values are available in the RNG_DR register, but it must not be used as described in *Section 23.3.5*.

To manage a seed error, follow the recommended sequence below. It requires the CONFIGLOCK bit to be cleared.

- When ARDIS is cleared (auto-reset enabled):
  - Clear the SEIS bit in the RNG_SR register
  - If SECS is cleared RNG can be used as usual. If SECS is still set in RNG_SR it means a new seed error occurred (unlikely). In this case, executes following step 1)
- When ARDIS is set (auto-reset disabled) the following sequence must be used to recover from a seed error.

1. Read both RNG_HTSRx registers. Using the previous HTSR values saved in RAM, compute the accumulated number of errors per oscillator (can be RPERR or ADERR errors). In a *nsmr* variable, mask the oscillators with accumulated number of errors greater than 3. This variable is used in step 3). Save the updated HTSR values in RAM.
   - During the first iteration, initialize *nsmr* variable with the RNG_NSCR value.
2. If fewer than 6 oscillators are active in the *nsmr* variable, reinitialize it with the RNG_NSCR value.
   - When RNG configuration is Configuration B, and the third time you reach this loop, go to Step 3 changing the RNG configuration to Configuration C. Both configurations are defined in *Section 23.6.2*.
3. Write CONDRST at 1 in RNG_CR, then update RNG_NSMR with *nsmr* value. Refer to *Section 23.7.8* for details.
4. Write CONDRST at 0 to finalize the reset of the RNG conditioning logic.
5. Wait for the CONDRST bit to be cleared by hardware, or poll for BUSY bit= 0. Then clear the SEIS interrupt status bit in the RNG_SR register.
6. If SECS is still set in RNG_SR it means a new seed error occurred (unlikely). In this case go back to step 1). If SECS is cleared, restart the generation of random numbers by setting RNGEN in RNG_CR.

If application has set the CONFIGLOCK bit (not recommended), do the following:

- When ARDIS is cleared (auto-reset enabled) clear the SEIS interrupt status bit in the RNG_SR register before drawing random numbers again.
- When ARDIS is set (auto-reset disabled), the following sequence must be used to fully recover from a seed error:

1. Write CONDRST at 1 and at 0 in the RNG_CR register to reset the conditioning logic.
2. Wait for the CONDRST bit to be cleared by hardware, or poll for BUSY = 0. Then clear the SEIS interrupt status bit in the RNG_SR register.
3. If SECS is still set in RNG_SR it means a new seed error occurred (unlikely). In this case go back to step 1).

Note: After a seed error RNG restarts generating random numbers when SECS is cleared.

RNG tamper errors

When an unexpected error is found by the RNG, an internal tamper event is triggered in the TAMP peripheral, and the RNG stops delivering random data.

When this event occurs, the secure application needs to reset the RNG peripheral, using the central reset management or the global SoC reset. A proper initialization of the RNG is required again.

Note: Seed errors do not trigger tamper errors.

23.3.8 RNG low-power use

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to 1 by setting the RNGEN bit to 0 in the RNG_CR register. As the post-processing logic and the output buffer remain operational while the RNGEN = 0, the following features are available to the software:

- If there are valid words in the output buffer, four random numbers can still be read from the RNG_DR register.
- If there are valid bits in the conditioning output internal register, four additional random numbers can be still be read from the RNG_DR register. If it is not the case, RNG must be re-enabled by the application until the expected new noise source bits threshold is reached (128-bit in NIST mode), and a complete conditioning round is done. Four new random words are then available only if the expected number of conditioning round is reached (two if NISTC = 0). The overall time can be found in *Section 23.5*.

When disabling the RNG, the user deactivates all the analog seed generators, whose power consumption is given in the datasheet electrical characteristics section. The user also gates all the logic clocked by the RNG clock. Note that this strategy is adding latency before a random sample is available on the RNG_DR register, because of the RNG initialization time.

If the RNG block is disabled during initialization (that is well before the DRDY bit rises for the first time), the initialization sequence resumes from where it was stopped when the RNGEN bit is set to 1, unless the application resets the conditioning logic using the CONDRST bit in the RNG_CR register.

When the application wants to gate the RNG kernel clock, the software must issue the following sequence: first, disable the RNG by clearing the RNGEN bit in the RNG_CR register, then wait for the BUSY bit to be cleared in the RNG_SR, and finally, disable the RNG kernel clock using the RCC.

Before entering a power mode where RNG is deactivated (like Stop mode), the software must first disable the RNG by clearing the RNGEN bit in the RNG_CR register, then wait for the BUSY bit to be cleared in the RNG_SR.

Note: The power modes where RNG is deactivated (retained or not available) can be found in the PWR section.

23.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see *Section 23.3.7*
- Clock error, see *Section 23.3.7*

Dedicated interrupt enable control bits are available as shown in *Table 178*.

Table 178. RNG interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
RNG	Data ready flag	DRDY	IE	None (automatic)
	Seed error flag	SEIS	IE	Write CONDRST to 1 then to 0 unless ARDIS is cleared (see *Section 23.3.7*). Write 0 to SEIS.
	Clock error flag	CEIS	IE	Write 0 to CEIS

The user can enable or disable the above interrupt sources globally by using the general interrupt control bit IE in the RNG_CR register. The status of the individual interrupt sources can be read from the RNG_SR register.

Note: Interrupts are generated only when RNG is enabled.

23.5 RNG processing time

In the recommended configuration described in *Table 179*, the time between two sets of four 32-bit data is either:

- 206 x N AHB cycles if $f_{AHB} < f_{threshold}$ (conditioning stage is limiting), or
- 128 x N RNG cycles $f_{AHB} \geq f_{threshold}$ (noise source stage is limiting).

If f_{RNG}= 48 MHz, $f_{threshold}$ = 1.6 x f_{RNG} = 77 MHz. The value of N is defined in *Section 23.6*.

Note: When CLKDIV is different from 0, f_{RNG} must take into account the internal divider ratio.

23.6 RNG entropy source validation

23.6.1 Introduction

To assess the amount of entropy available from the RNG, the peripheral has been tested using the German BSI AIS-31 statistical tests (T0 to T8), and NIST SP800-90B test suite.

23.6.2 Validation conditions

The RNG true random number generator has been tested in the following conditions:

- RNG clock rng_clk = 48 MHz, with CED bit cleared in RNG_CR.
- Configurations are described in *Table 179* and *Table 180*. Only configuration A can be certified NIST SP800-90B. Refer to *Table 181* to select the best configuration for the application.
- Configuration B or C can be used when configuration A is not present in AN4230, or is not flagged as certified.

Table 179. RNG configurations

Configuration	RNG_CR bits					Loop number (N)	RNG_HTCR0 register	RNG_NSCR register
	NISTC bit	RNG_CONFIG1 [7:0]	CLKDIV [3:0]	RNG_CONFIG2 [2:0]	RNG_CONFIG3 [3:0](1)			
A	Refer to *NIST compliant RNG configuration* table in AN4230 available from *www.st.com*. This application note also indicates if this configuration is part of an existing NIST SP800-90B Entropy Certificate listed on *https://csrc.nist.gov/projects/cryptographic-module-validation-program.*							
B	1	0x83	0x0	0x0	0xF	3	0x0000 AAC7(2)	Default
C	1	0x84	0x5(3)	0x0	0xF	4		Default

1. RNG_CONFIG1[3:0] = i defines the loop number N. If i = 0, N =1. If NISTC = 0, N = 2 regardless of i.
2. This value can be fixed in the RNG driver (it does not depend upon the STM32 product).
3. The noise source sampling must be 48 MHz or less. If the RNG clock is different from 48 MHz, this value of CLKDIV must be adapted. See the CLKDIV bitfield description in *Section 23.7.1* for details.

Table 180. Additional health test configurations

Configuration	RNG_HTCR1 register (oscillator set 1)	RNG_HTCR2 register (oscillator set 2)	RNG_HTCR3 register (oscillator set 3)
A	Refer to NIST compliant RNG configuration table in AN4230 available from www.st.com.		
B and C	default	default	default

Table 181. Configuration selection

Section criteria	Configuration A	Configuration B	Configuration C
Suitable to generate NIST compliant cryptographic keys	Yes	No	
Entropy(1)	Certified	Not certified	
Speed(2)	Baseline	Faster	Baseline

1. For configurations B and C entropy is verified using the AIS-31 test suite (T0 to T8).
2. When the speed is not enough for the application, an NIST compliant DRBG can be used to increase throughput.

For details on data collection and the running of statistical test suites refer to AN4230 "*Introduction to random number generation validation using the NIST statistical test suite for STM32 MCUs and MPUs*", available on *www.st.com*.

23.7 RNG registers

The RNG is associated with a control register, a data register, and a status register.

23.7.1 RNG control register (RNG_CR)

Address offset: 0x000

Reset value: 0x0080 0D00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CONFIGLOCK	CONDRST	Res.	Res.	RNG_CONFIG1[7:0]								CLKDIV[3:0]			
rs	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNG_CONFIG2[2:0]			NISTC	RNG_CONFIG3[3:0]				ARDIS	Res.	CED	Res.	IE	RNGEN	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw		

Bit 31 **CONFIGLOCK**: RNG configuration lock

Set this bit to lock the RNG configuration. Since CONFIGLOCK prevents the implementation of robust seed error management described in *Section 23.3.7*, it is recommended not to use it.

0: Writes to the RNG_NSMR, RNG_NSCR, RNG_HTCR and the RNG_CR configuration bits [29:4] are allowed.

1: Writes to the RNG_NSMR, RNG_NSCR, RNG_HTCR and the RNG_CR configuration bits [29:4] are ignored until the next RNG reset.

Once set, this bit can only be cleared when RNG is reset (set once bit).

Bit 30 **CONDRST**: Conditioning soft reset

Write 1 and then write 0 to reset the conditioning logic, clear all the FIFOs and start a new RNG initialization process, with RNG_SR cleared. Registers RNG_NSMR, RNG_NSCR, RNG_CR and RNG_HTCR are not changed by CONDRST.

This bit must be set to 1 in the same access that sets any configuration bits [29:4]. In other words, when CONDRST bit is set to 1 correct configuration in bits [29:4] must also be written.

Before setting this bit to 1 RNGEN must be set to 0, and BUSY flag must be cleared in RNG_SR.

When CONDRST is set to 0, wait for BUSY flag to be cleared in RNG_SR. CONDRST bit then reads as 0.

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:20 **RNG_CONFIG1[7:0]:** RNG configuration 1

Reserved to the RNG configuration (bitfield 1). Must be initialized using the recommended value documented in *Section 23.6*.

Writing to this bitfield is only effective when CONFIGLOCK = 0. The new value is only taken into account if CONDRST bit is set in the same access.

Bits 19:16 **CLKDIV[3:0]**: Clock divider factor

This value used to configure an internal programmable divider (from 1 to 16) acting on the incoming RNG clock. These bits can be written only when the core is disabled (RNGEN = 0).

0x0: internal RNG clock after divider is similar to incoming RNG clock.

0x1: two RNG clock cycles per internal RNG clock.

0x2: 2^2 (= 4) RNG clock cycles per internal RNG clock.

...

0xF: 2^{15} RNG clock cycles per internal clock (for example. an incoming 48 MHz RNG clock becomes a 1.5 kHz internal RNG clock)

Writing to this bitfield is only effective when CONFIGLOCK = 0. The new value is only taken into account if CONDRST bit is set in the same access.

Bits 15:13 **RNG_CONFIG2[2:0]:** RNG configuration 2

Reserved to the RNG configuration (bitfield 2). Refer to the RNG_CONFIG1 bitfield for details.

Bit 12 **NISTC**: NIST custom

0: Hardware default values for NIST compliant RNG. In this configuration for 128-bit output, two conditioning loops are performed and 256 bits of noise source are used.

1: Custom values for NIST compliant RNG. See *Section 23.6* for recommended configurations.

Writing to this bit is only effective when CONFIGLOCK = 0. The new value is only taken into account if the CONDRST bit is set in the same access.

Bits 11:8 **RNG_CONFIG3[3:0]:** RNG configuration 3

Reserved to the RNG configuration (bitfield 3). Refer to RNG_CONFIG1 bitfield for details.

If the NISTC bit is cleared in this register RNG_CONFIG3 bitfield values are ignored by RNG.

Bit 7 **ARDIS**: Auto reset disable

Set this bit to deactivate the auto-reset feature.

0: Auto-reset is enabled

1: Auto-reset is disabled

Keeping the auto-reset enabled (automatic clearance of the SECS bit) simplifies the management of noise source errors, as described in *Section 23.3.7*.

Writing to this bit is only effective when CONFIGLOCK = 0. The new value is only taken into account if the CONDRST bit is set in the same access.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CED:** Clock error detection

0: Clock error detection is enabled.

1: Clock error detection is disabled

The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, that is to enable or disable CED, the RNG must be disabled.

Writing to this bit is only effective when CONFIGLOCK = 0. The new value is only taken into account if the CONDRST bit is set in the same access.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **IE:** Interrupt enable

0: RNG interrupt is disabled

1: RNG interrupt is enabled. An interrupt is pending as soon as the DRDY, SEIS, or CEIS flag is set in the RNG_SR register.

Bit 2 **RNGEN:** True random number generator enable
This bit enables/disables the true random number generator.
0: True random number generator disabled.
1: True random number generator enabled.
Clearing RNGEN powers off the analog noise sources and gates the logic clocked by the RNG clock.
When clearing this bit verify that BUSY flag is cleared in RNG_SR before updating this register again.

Bits 1:0 Reserved, must be kept at reset value.

23.7.2 RNG status register (RNG_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	BUSY	Res.	SECS	CECS	DRDY
									rc_w0	rc_w0	r		r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SEIS:** Seed error interrupt status
This bit is set at the same time as SECS. It is cleared by writing 0 (unless CONDRST is used). Writing 1 has no effect.
0: No faulty sequence detected
1: At least one faulty sequence is detected. See SECS bit description for details.
An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 5 **CEIS:** Clock error interrupt status
This bit is set at the same time as CECS. It is cleared by writing 0. Writing 1 has no effect.
0: The RNG clock is correct ($f_{RNGCLK} > f_{HCLK}/32$)
1: The RNG clock before the internal divider is detected too slow ($f_{RNGCLK} < f_{HCLK}/32$)
An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 4 **BUSY:** Busy
This flag indicates whether RNG is idle or busy.
0: Idle
1: Busy
RNG is flagged as busy when a conditioning soft reset (using CONDRST) is in progress. It is also flagged busy when RNG is activated by application setting RNGEN bit, or when a connected peripheral fetches randoms via the dedicated RNG hardware bus.
More information on BUSY usage can also be found in *Section 23.3.8*.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SECS:** Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.

1: At least one of the following faulty sequence has been detected:

– Start-up or continuous repetition count test on noise source failed.

– Start-up or continuous adaptive proportion test on noise source failed.

– Start-up post-processing/conditioning sanity check failed.

After enabling the RNG, SECS bit clears once startup health tests completes successfully. If SECS does not clear, run the software reset sequence with CONDRST described in *Section 23.3.4*.

Bit 1 **CECS:** Clock error current status

0: The RNG clock is correct ($f_{RNGCLK} > f_{HCLK}/32$). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.

1: The RNG clock is too slow ($f_{RNGCLK} < f_{HCLK}/32$).

CECS bit is valid only if the CED bit in the RNG_CR register is set to 0.

Bit 0 **DRDY:** Data ready

This flag is raised to indicate valid (non-0) random data available in the RNG_DR register. It is cleared by hardware upon reading the RNG_DR register (which empties the RNG buffer). The DRDY flag can also be raised upon disabling RNG (upon clearing the RNGEN bit).

0: No valid random data ready.

1: Valid random data ready.

Raising DRDY generates an interrupt if enabled with the IE bit of the RNG_CR register.

23.7.3 RNG data register (RNG_DR)

Address offset: 0x008

Reset value: 0x0000 0000

This register is a read-only register that delivers a 32-bit random value when read. The content of this register is valid when the DRDY = 1 and the value is not 0x0, even if RNGEN = 0.

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

Bits 31:0 **RNDATA[31:0]:** Random data

32-bit random data, valid when DRDY = 1. When DRDY = 0, the RNDATA value is 0.

The 0 value means that a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

23.7.4 RNG noise source control register (RNG_NSCR)

Address offset: 0x00C

Reset value: 0x0000 01FF

Writing in RNG_NSCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in RNG_CR. Writing to this register is ignored if CONFIGLOCK= 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_OSC3[2:0]			EN_OSC2[2:0]			EN_OSC1[2:0]		
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:6 **EN_OSC3[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 3. The bitfield has no effect otherwise.

Bits 5:3 **EN_OSC2[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 2. The bitfield has no effect otherwise.

Bits 2:0 **EN_OSC1[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 1. The bitfield has no effect otherwise.

23.7.5 RNG health test control register x (RNG_HTCRx)

Address offset: 0x010 + 0x4 * x (x=0 to 3)

Reset value: 0x0000 72AC, 0x0003 FFFF, 0x0003 FFFF, 0x0003 FFFF

Writing in RNG_HTCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in the RNG_CR. Writing to this register is ignored if CONFIGLOCK = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HTCFG[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTCFG[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HTCFG[31:0]:** health test configuration

This configuration is used by RNG to configure the health tests (repetition count test, adaptive proportion test) at position x. See *Section 23.6* for the recommended value. With its default value, health tests have a very small probability of triggering any error at position x.

Note: The RNG behavior, including the read to this register, is not guaranteed if a different value from the recommended value is written.

23.7.6 RNG health test status register 0 (RNG_HTSR0)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RPERR9	RPERR8	RPERR7	RPERR6	RPERR5	RPERR4	RPERR3	RPERR2	RPERR1	RPERRX
						r	r	r	r	r	r	r	r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:1 **RPERR{i}:** Repetitive error for oscillator *i* (i=1 to 9)

This bit is set when the health test logic associated to the oscillator *i* finds a repetitive test error. The bit is cleared when application clears SEIS bit in RNG_SR. If ARDIS is cleared auto-reset prevents RNG to capture more than one test error. Repetition count test is defined in *Health checks* section.

0: No repetitive test error found in oscillator *i*.
1: Repetitive test error found in oscillator *i*.

Bit 0 **RPERRX:** Repetitive error after the XOR

This bit is set when the health test logic finds a repetitive test error after the XOR clock tree (see *Figure 109*). The bit is cleared when application clears SEIS bit in RNG_SR. If ARDIS is cleared auto-reset prevents RNG to capture more than one test error.

0: No repetitive test error found after the XOR clock tree.
1: Repetitive test error found after the XOR clock tree.

23.7.7 RNG health test status register 1 (RNG_HTSR1)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	ADERR9	ADERR8	ADERR7	ADERR6	ADERR5	ADERR4	ADERR3	ADERR2	ADERR1	ADERRX
						r	r	r	r	r	r	r	r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:1 **ADERR{i}:** Adaptative error for oscillator *i* (i=1 to 9)

This bit is set when the health test logic associated to the oscillator *i* finds an adaptative test error. The bit is cleared when application clears SEIS bit in RNG_SR. If ARDIS is cleared auto-reset prevents RNG to capture more than one test error. Adaptive proportion test is defined in *Health checks* section.

0: No adaptative test error found in oscillator *i*.
1: Adaptative test error found in oscillator *i*.

Bit 0 **ADERRX:** Adaptative error after the XOR

This bit is set when the health test logic finds an adaptative test error after the XOR clock tree (see *Figure 109*). The bit is cleared when application clears SEIS bit in RNG_SR If ARDIS is cleared auto-reset prevents RNG to capture more than one test error.

0: No adaptative test error found after the XOR clock tree.
1: Adaptative test error found after the XOR clock tree.

23.7.8 RNG noise source mask register (RNG_NSMR)

Address offset: 0x030

Reset value: 0x0000 01FF

Writing to this register is ignored if CONFIGLOCK = 1. It is recommended to have at least 6 oscillators running when loop number N = 3 (or higher). See *Section 23.6.2* for details.

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

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **MOSC{i}:** Mask oscillator i (i = 1 to 9)

Clear this bit to masks the oscillator *i*, and deactivate its related health tests. The bit has no effect if the oscillator *i* is deactivated with the bit RNG_NSCR[i].

Application can mask an oscillator when it is not contributing to the generation of entropy.

0: Oscillator *i* is masked, and related health tests are deactivated.
1: Oscillator *i* is not masked. Related health tests are running.

23.7.9 RNG register map

Table 182. RNG register map and reset map

Offset	Register name	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
0x000	RNG_CR	CONFIGLOCK	CONDRST	Res.	Res.	RNG_CONFIG1[7:0]								CLKDIV[3:0]				RNG_ CONFIG2 [2:0]			NISTC	RNG_ CONFIG3 [3:0]				ARDIS	Res.	CED	Res.	IE	RNGEN	Res.	Res.
	Reset value	0	0			0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0		0		0	0		
0x004	RNG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	BUSY	Res.	SECS	CECS	DRDY
	Reset value																										0	0	0		0	0	0
0x008	RNG_DR	RNDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	RNG_NSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_OSC3 [2:0]		EN_OSC2 [2:0]			EN_OSC1 [2:0]			
	Reset value																								1	1	1	1	1	1	1	1	1
0x010	RNG_HTCR0	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	1	0	1	1	0	0
0x014	RNG_HTCR1	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x018	RNG_HTCR2	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x01F	RNG_HTCR3	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x020	RNG_HTSR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPERR9	RPERR8	RPERR7	RPERR6	RPERR5	RPERR4	RPER3	RPERR2	RPERR1	RPERRX
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x024	RNG_HTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADERR9	ADERR8	ADERR7	ADERR6	ADERR5	ADERR4	ADERR3	ADERR2	ADERR1	ADERRX
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x030	RNG_MSMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MOSC9	MOSC8	MOSC7	MOSC6	MOSC5	MOSC4	MOSC3	MOSC2	MOSC1
	Reset value																								1	1	1	1	1	1	1	1	1

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

24 AES hardware accelerator (AES)

24.1 AES introduction

The AES hardware accelerator (AES) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST.

AES supports ECB, CBC, CTR, GCM, GMAC, and CCM chaining modes for key sizes of 128 or 256 bits.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels are required).

24.2 AES main features

- Compliant with NIST FIPS publication 197 *"Advanced encryption standard (AES)"* (November 2001)
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- 128-bit data block processing, supporting cipher key lengths of 128-bit and 256-bit
  - 51 or 75 clock cycle latency in ECB mode for processing one 128-bit block with, respectively, 128-bit or 256-bit key
- Integrated key scheduler to compute the last round key for ECB/CBC decryption
- 256-bit of write-only registers for storing cryptographic keys (eight 32-bit registers)
- 128-bit of registers for storing initialization vectors (four 32-bit registers)
- 32-bit buffer for data input and output
- Automatic data flow control supporting two direct memory access (DMA) channels, one for incoming data, one for processed data. Only single transfers are supported.
- Data-swapping logic to support 1-, 8-, 16-, or 32-bit data
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.
- Possibility for software (in CPU mode only, not in DMA mode) to suspend a message if AES needs to process another message with a higher priority, then resume the original message

24.3 AES implementation

The devices have one AES peripheral, implemented as per the following table.

Table 183. AES features

Modes or features	AES(1)
ECB, CBC chaining	X
CTR, CCM, GCM chaining	X
AES 128-bit ECB encryption in cycles	51
DHUK and BHK key selection	-

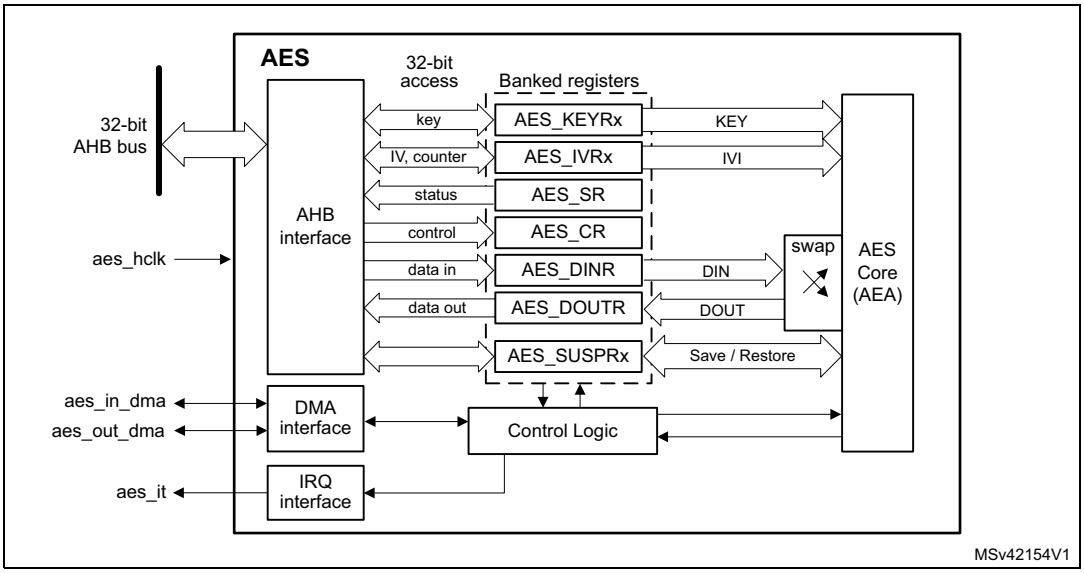
1. X = supported.

24.4 AES functional description

24.4.1 AES block diagram

Figure 112 shows the block diagram of AES.

Figure 112. AES block diagram



24.4.2 AES internal signals

Table 184 describes the user relevant internal signals interfacing the AES peripheral.

Table 184. AES internal input/output signals

Signal name	Signal type	Description
aes_hclk	Input	AHB bus clock
aes_it	Output	AES interrupt request
aes_in_dma	Input/Output	AES incoming data DMA single request/acknowledge

Table 184. AES internal input/output signals (continued)

Signal name	Signal type	Description
aes_out_dma	Input/Output	AES processed data DMA single request/acknowledge
aes_itamp_out	Output	Tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, AES automatically clears key registers. A reset is required for AES to be usable again.

24.4.3 AES reset and clocks

The AES peripheral is clocked by the AHB bus clock. It has a dedicated reset bit controlled through the RCC.

24.4.4 AES symmetric cipher implementation

The AES hardware accelerator (AES) is a 32-bit AHB peripheral that encrypts or decrypts 16-byte blocks of data using the advanced encryption standard (AES). It also implements a set of approved AES symmetric key security functions summarized in *Table 185*. Those functions can be certified NIST PUB 140-3.

Table 185. AES approved symmetric key functions

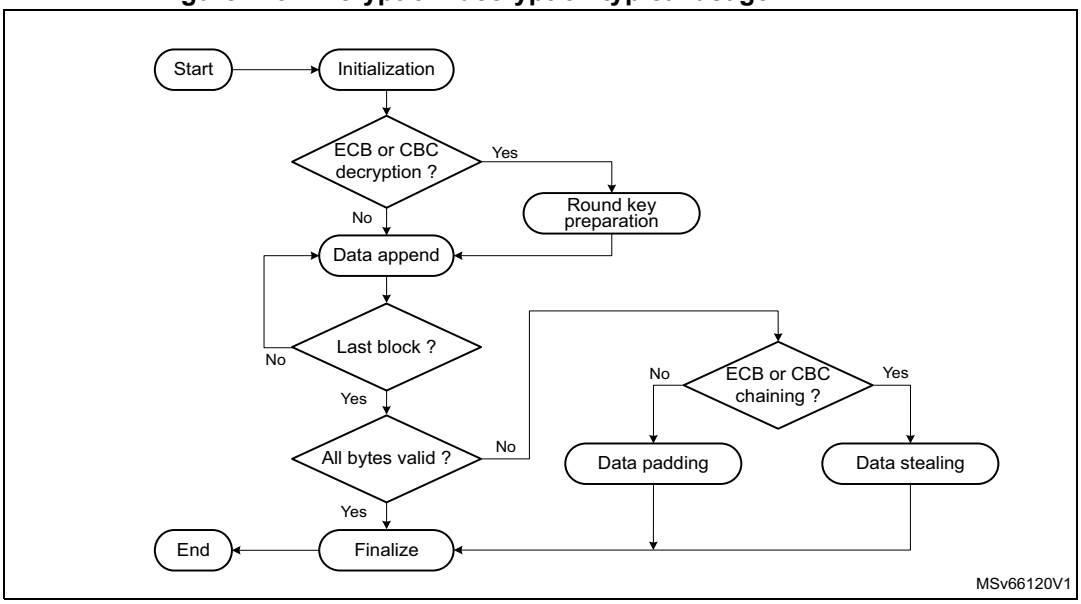
Operations	Algorithm	Specification	Key bit lengths	Chaining modes
Encryption, decryption	AES	FIPS PUB 197 NIST SP800-38A	128, 256	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D		GCM, CCM
Cipher-based message authentication code		NIST SP800-38D		GMAC

AES can be used directly by the CPU, or indirectly, using two DMA channels (one for the plaintext, one for the ciphertext).
It is possible to suspend then resume any AES processing, following the sequence described in *Section 24.4.8*.

24.4.5 AES encryption or decryption typical usage

The following figure shows a typical operation for encryption or decryption.

Figure 113. Encryption/ decryption typical usage



Initialization

The AES peripheral is initialized according to the chaining mode. Refer to *Section 24.4.9: AES basic chaining modes (ECB, CBC)* and *Section 24.4.10: AES counter (CTR) mode* for details.

Data append

This section describes different ways of appending data for processing. For ECB or CBC chaining modes, refer to *Section 24.4.7: AES ciphertext stealing and data padding* if the size of data to process is not a multiple of 16 bytes. The last block management in these cases is more complex than what is described in this section.

Appending data using the CPU in polling mode

This method uses flag polling to control the data append through the following sequence:

1. Enable the AES peripheral when KEYVALID is set, by setting the EN bit of the AES_CR register (if not already done).
2. Repeat the following sub-sequence until the payload is entirely processed:
   a) Write four input data words into the AES_DINR register.
   b) Wait until the status flag CCF is set in the AES_ISR register, then read the four data words from the AES_DOUTR register.
   c) Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
   d) If the next processing block is the last block, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (using

NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong).

3. As the data block just processed is the last block of the message, optionally discard the data that is not part of the message/payload, then disable the AES peripheral by clearing EN.

Note: Up to three wait cycles are automatically inserted between two consecutive writes to the AES_DINR register, to allow sending the key to the AES processor.

NPBLB[3:0] bitfield is not used in header phase of GCM, GMAC and CCM chaining modes.

Appending data using the CPU in interrupt mode

The method uses interrupt from the AES peripheral to control the data append, through the following sequence:

1. Enable interrupts from AES, by setting the CCFIE bit of the AES_IER register.
2. Enable the AES peripheral when KEYVALID is set, by setting EN (if not already done).
3. Write first four input data words into the AES_DINR register.
4. Handle the data in the AES interrupt service routine. Upon each interrupt:
   a) Read four output data words from the AES_DOUTR register.
   b) Clear the CCF flag and thus the pending interrupt, by setting the CCF bit of the AES_ICR register.
   c) If the next processing block is the last block of the message, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (through NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). Then proceed with point 4e).
   d) If the data block just processed is the last block of the message, optionally discard the data that are not part of the message/payload, then disable the AES peripheral by clearing EN and quit the interrupt service routine.
   e) Write next four input data words into the AES_DINR register and quit the interrupt service routine.

Note: AES is tolerant of delays between consecutive read or write operations, which allows, for example, an interrupt from another peripheral to be served between two AES computations.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

Appending data using DMA

With this method, all the transfers and processing are managed by DMA and AES. Proceed as follows:

1. If the last block of the message to process is shorter than 16 bytes, prepare the last four-word data block by padding the remainder of the block with zeros.
2. Configure the DMA controller so as to transfer the data to process from the memory to the AES peripheral input and the processed data from the AES peripheral output to the memory, as described in *Section 24.7: AES DMA requests*. Configure the DMA controller so as to generate an interrupt on transfer completion. For GCM payload encryption or CCM payload decryption, the DMA transfer **must not** include the last four-word block if padded with zeros. The sequence described in *Appending data using the CPU in polling mode* must be used instead for this last block, because the

NPBLB[3:0] bitfield must be set up before processing the block, for AES to compute a correct tag.

3. Enable the AES peripheral when KEYVALID is set, by setting EN (if not already done).
4. Enable DMA requests, by setting DMAINEN and DMAOUTEN.
5. Upon DMA interrupt indicating the transfer completion, get the AES-processed data from the memory.

When appending data using DMA, the suspend/resume operation as described in *Section 24.4.8* is not supported.

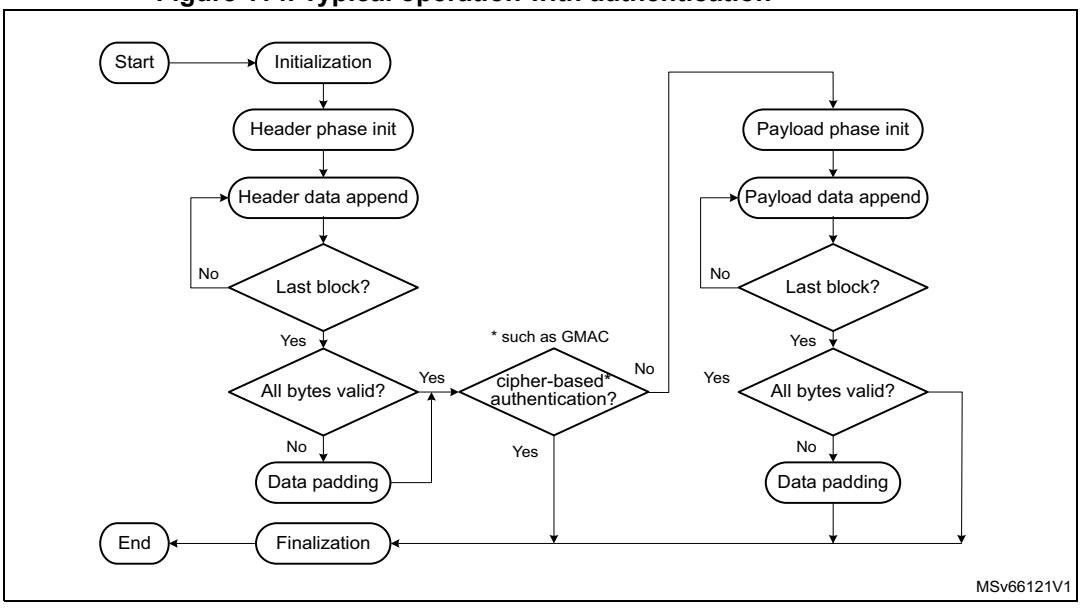
Note: The CCF flag has no use with this method because the reading of the AES_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

24.4.6 AES authenticated encryption, decryption, and cipher-based message authentication

The following figure shows a typical operation for authenticated encryption or decryption, and for cipher-based message authentication.

Figure 114. Typical operation with authentication



Section 24.4.11: AES Galois/counter mode (GCM) and *Section 24.4.13: AES counter with CBC-MAC (CCM)* describe detailed sequences supported by AES.

Cipher-based message authentication flow omits the payload phase, as shown in the figure. Detailed sequence supported by AES is described in *Section 24.4.12: AES Galois message authentication code (GMAC)*.

24.4.7 AES ciphertext stealing and data padding

When using AES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (16 bytes), the application must use ciphertext stealing techniques

such as those described in NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. Since AES does not implement such techniques, the application must complete the *last block* of input data using data from the *second last* block.

Note: Ciphertext stealing techniques are not documented in this reference manual.

Similarly, in modes other than ECB or CBC, an incomplete input data block (that is, a block with input data shorter than 16 bytes) must be padded with zeros prior to encryption. That is, extra bits must be appended to the trailing end of the data string. After decryption, the extra bits must be discarded. Since AES does not implement automatic data padding operation to the *last block*, the application must follow the recommendation given in this document to manage messages the size of which is not a multiple of 16 bytes.

24.4.8 AES suspend and resume operations

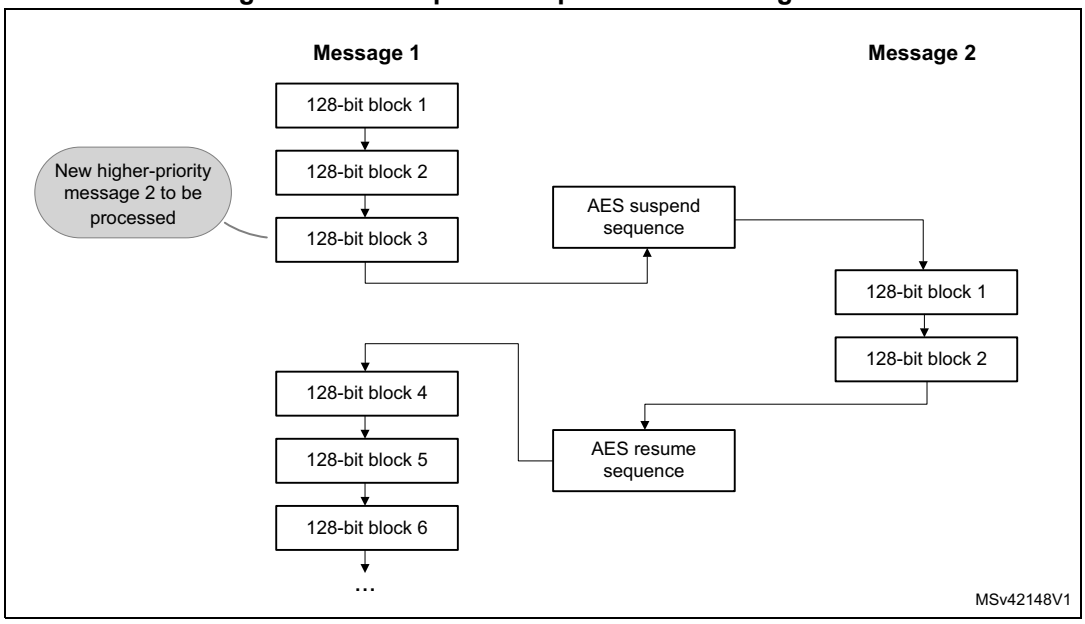
A message can be suspended to process another message with a higher priority. When the higher-priority message is sent, the suspended message can resume. This applies to both encryption and decryption mode.

Suspend and resume operations do not break the chaining operation. The message processing can resume as soon as AES is enabled again, to receive a next data block.

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

Figure 115 gives an example of suspend and resume operations: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 115. Example of suspend mode management



A detailed description of suspend and resume operations is in the sections dedicated to each chaining mode.

24.4.9 AES basic chaining modes (ECB, CBC)

ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately. When decrypting in ECB, a special key scheduling is required before processing the first block.

Figure 116 and *Figure 117* describe the electronic codebook (ECB) chaining implementation in encryption and in decryption, respectively. To select ECB chaining mode, write CHMOD[2:0] with 0x0.

Figure 116. ECB encryption

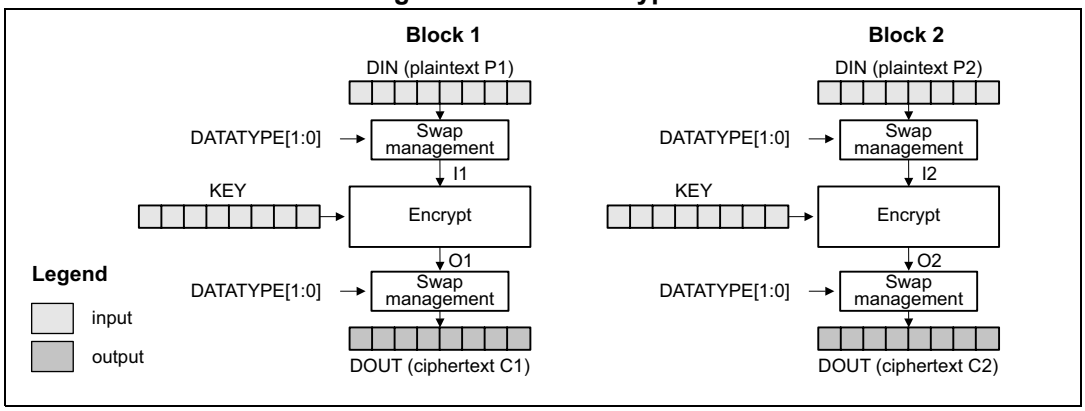
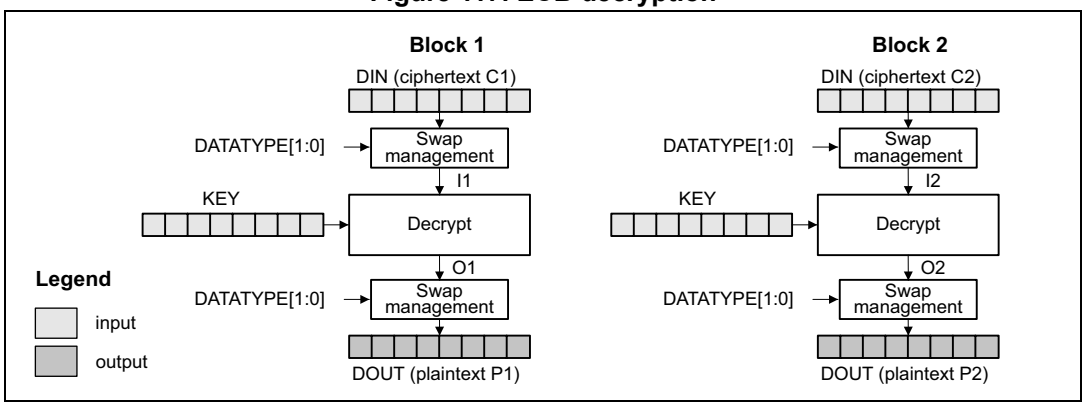


Figure 117. ECB decryption



In CBC encryption mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing. When decrypting in CBC, a special key scheduling is required before processing the first block.

Figure 118 and *Figure 119* describe the cipher block chaining (CBC) implementation in encryption and in decryption, respectively. To select this chaining mode, write CHMOD[2:0] with 0x1.

Figure 118. CBC encryption

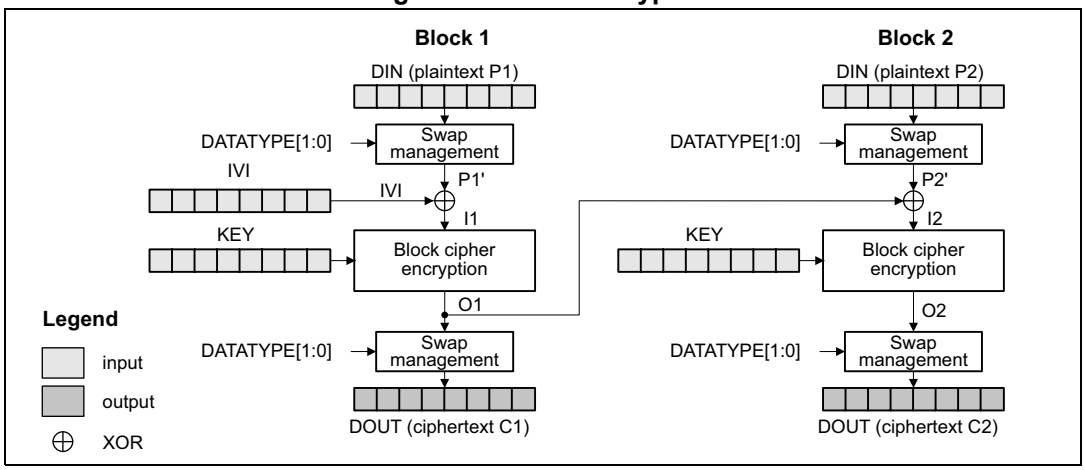
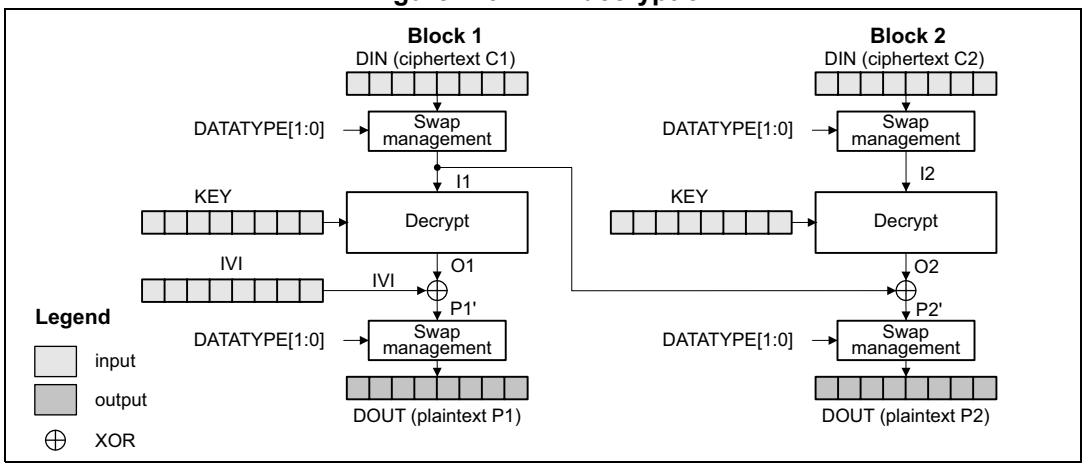


Figure 119. CBC decryption



For more details, refer to NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation.*

ECB and CBC encryption process

This process is described in *Section 24.4.5*, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register as follows:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *encryption* mode (write MODE[1:0] with 0x0).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
3. Write the initialization vector into the AES_IVRx registers if CBC mode is selected in the previous step.
4. Write the key into the AES_KEYRx registers.

5. Wait until KEYVALID is set (the key loading completed).
6. Enable the AES peripheral, by setting EN.
7. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 24.4.7*.
   b) Append the cleartext block into AES as described in *Section 24.4.5*, then read the AES_DOUTR register four times to save the ciphertext block.
   c) Repeat the step *b)* until the third-last plaintext block is encrypted. For the last two blocks, follow the steps *a)* and *b)*.
8. Finalize the sequence: disable the AES peripheral, by clearing EN.

ECB/CBC decryption process

This process is described in *Section 24.4.5*, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register as follows:
   – Select the *key derivation* mode (write MODE[1:0] with 0x1). The CHMOD[2:0] bitfield is not significant during this operation.
   – Configure the data type, through DATATYPE[1:0].
   – Configure the key size, through KEYSIZE.
3. Write the key into the AES_KEYRx registers.
4. Wait until KEYVALID is set (the key loading completed).
5. Enable the AES peripheral, by setting EN. The peripheral immediately starts an AES round for key preparation.
6. Wait until the CCF flag in the AES_ISR register is set.
7. Clear the CCF flag, by setting the CCF bit of the AES_ICR register. The decryption key is available in the AES core and AES is disabled automatically.
8. Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *decryption* mode (write MODE[1:0] with 0x2). Do not change other parameters.
9. Write the initialization vector into the AES_IVRx registers if CBC mode is selected in the previous step.
10. Enable the AES peripheral, by setting EN.
11. Append encrypted data:
    a) If it is the second-last or the last block and the ciphertext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 24.4.7*.
    b) Append the ciphertext block into AES as described in *Section 24.4.5*, then read the AES_DOUTR register four times to save the cleartext block (MSB first).
    c) Repeat the step *b)* until the third-last ciphertext block is decrypted. For the last two blocks, follow the steps *a)* and *b)*.
12. Finalize the sequence: disable the AES peripheral, by clearing EN.

Suspend/resume operations in ECB/CBC modes

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

To suspend the processing of a message, proceed as follows:

1. Wait until the CCF flag in the AES_ISR register is set (computation completed).
2. Read four times the AES_DOUTR register to save the last processed block.
3. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
4. Save initialization vector registers (only required in CBC mode as the AES_IVRx registers are altered during the data processing).
5. Disable the AES peripheral, by clearing EN.
6. Save the AES_CR register and clear the key registers if they are not needed, to process the higher-priority message.

To resume the processing of a message, proceed as follows:

1. Disable the AES peripheral, by clearing EN.
2. Restore the AES_CR register (with correct KEYSIZE) then restore the AES_KEYRx registers.
3. Prepare the decryption key, as described in *ECB/CBC decryption process* (only required for ECB or CBC decryption).
4. Restore the AES_IVRx registers, using the saved configuration (only required in CBC mode).
5. Enable the AES peripheral, by setting EN.

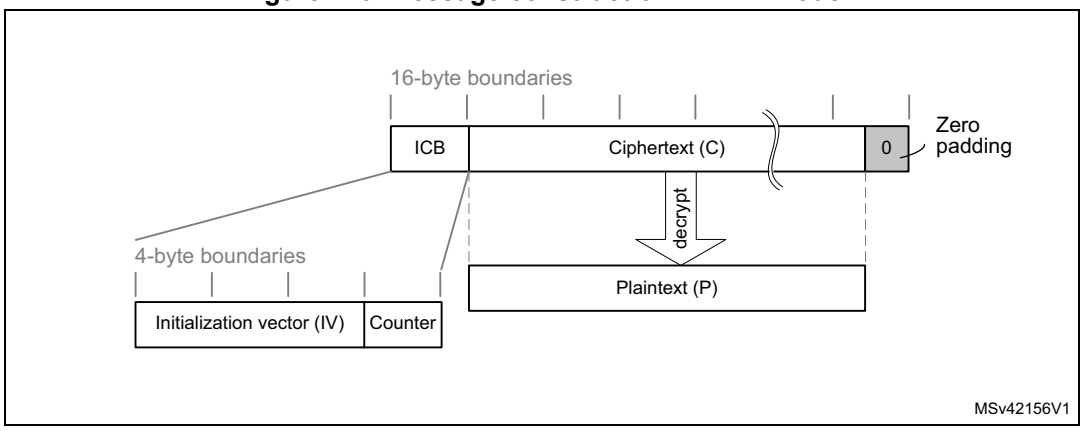
Note: It is not required to save the key registers as the application knows the original key.

24.4.10 AES counter (CTR) mode

The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext. Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption since the AES core is always used in encryption mode.

A typical message construction in CTR mode is given in *Figure 120*.

Figure 120. Message construction in CTR mode



The structure of this message is:

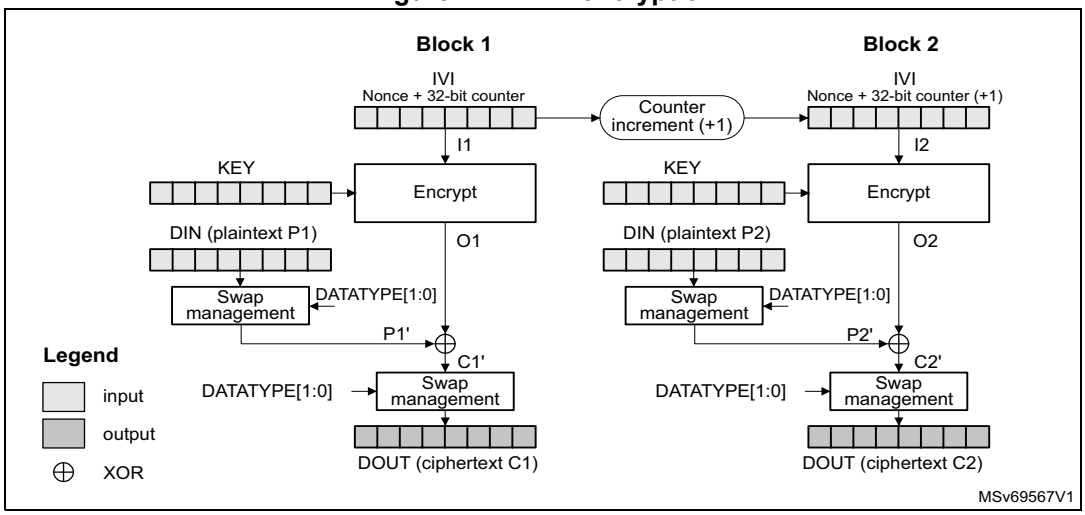
- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IV): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. The initial value of the counter must be set to 1.
- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.*

CTR encryption and decryption

Figure 121 describes the counter (CTR) chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x2.

Figure 121. CTR encryption



Initialization vectors in AES must be initialized as shown in *Table 186*.

Table 186. Counter mode initialization vector definition

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0] 32-bit counter = 0x0001

CTR encryption and decryption process

This process is described in *Section 24.4.5*, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select CTR chaining mode (write CHMOD[2:0] with 0x2) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.

3. Write the initialization vector into the AES_IVRx registers according to *Table 186*.
4. Write the key into the AES_KEYRx registers.
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the AES peripheral, by setting EN.
7. Append data:
   a) If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is less than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into AES as described in *Section 24.4.5*, then read the AES_DOUTR register four times to save the resulting block (MSB first).
   c) Repeat the step *b)* until the second-last block is processed. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the message when the last block is smaller than 16 bytes.
8. Finalize the sequence: disable the AES peripheral, by clearing EN.

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher-priority message, then resume the interrupted message. Detailed CBC suspend and resume sequence is described in *Section 24.4.9: AES basic chaining modes (ECB, CBC)*.

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

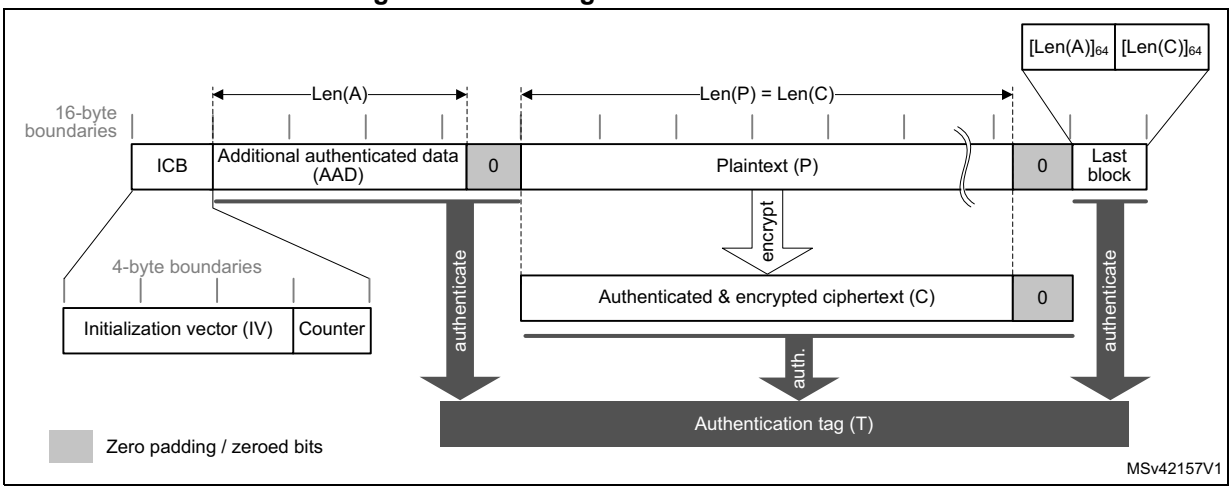
Note: Like for CBC mode, the IV registers must be reloaded during the resume operation.

24.4.11 AES Galois/counter mode (GCM)

The AES Galois/counter mode (GCM) allows encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code).

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. The following figure shows a typical message construction in GCM mode.

Figure 122. Message construction in GCM



The message has the following structure:

- **16-byte initial counter block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV):** a 96-bit value that must be unique for each encryption cycle with a given key. The GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter:** a 32-bit big-endian integer that is incremented each time a block processing is completed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- **Authenticated header AAD** (also knows as additional authentication data) has a known length Len(A) that may be a non-multiple of 16 bytes, and must not exceed $2^{64} – 1$ bits. This part of the message is only authenticated, not encrypted.
- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length Len(P) that may be non-multiple of 16 bytes, and cannot exceed 2^{32} - 2 16-byte blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in *Table 188*.

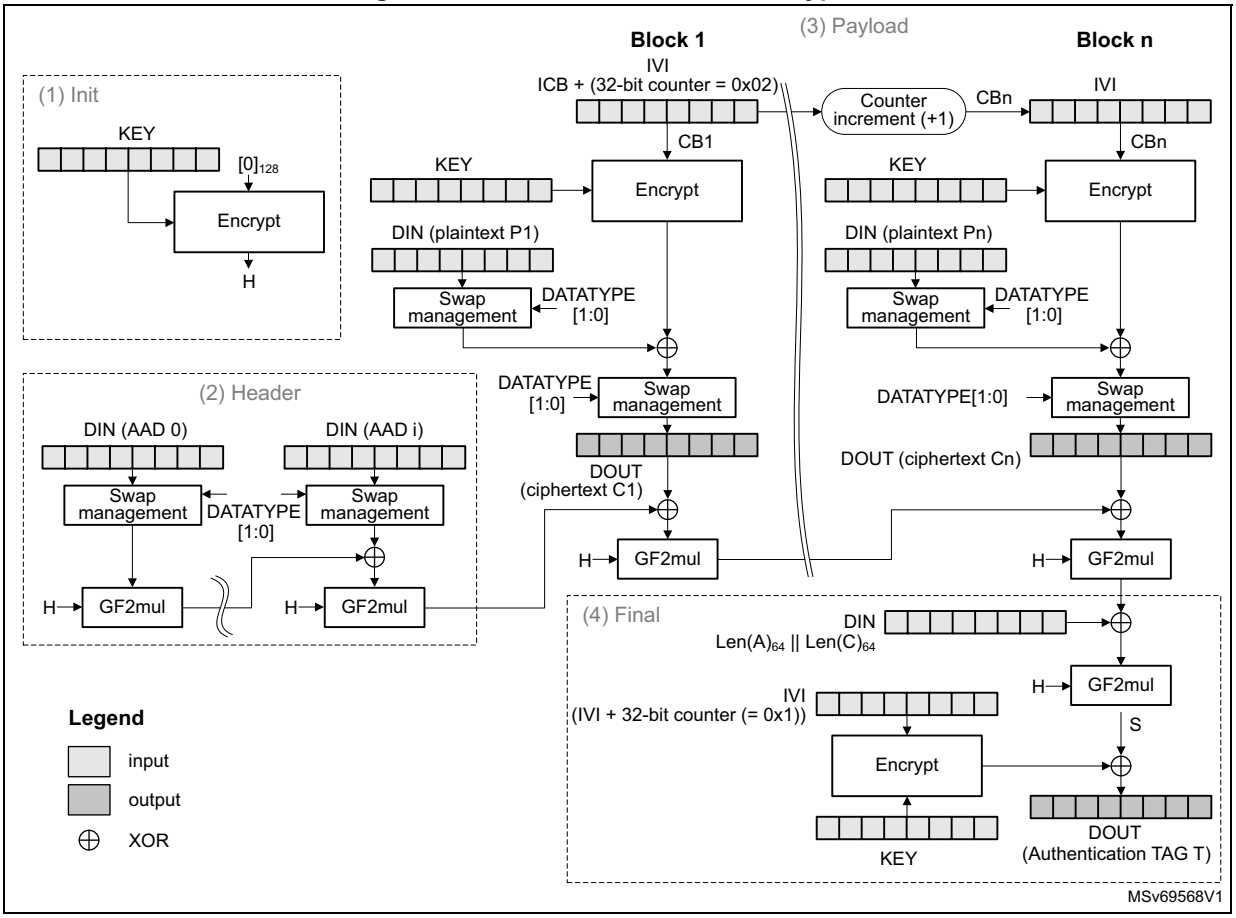
The GCM standard specifies that ciphertext C has the same bit length as the plaintext P.

When a part of the message (AAD or P) has a length that is a non-multiple of 16-bytes a special padding scheme is required.

For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 123 describes the GCM chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 123. GCM authenticated encryption



The first counter block (CB1) is derived from the initial counter block ICB by the application software, as defined in *Table 187*.

Table 187. Initialization of IV registers in GCM mode

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x0002

The last block of a GCM message contains the AAD header length and the payload length information, as shown in *Table 188*.

Table 188. GCM last block definition

Word order to AES_DINR	First word	Second word	Third word	Fourth word
Input data	AAD length[63:32]	AAD length[31:0]	Payload length[63:32]	Payload length[31:0]

GCM encryption and decryption process

This process is described in *Section 24.4.6*, with the following sequence of events:

GCM initialize

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select GCM chaining mode (write CHMOD[2:0] with 0x3) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE.
   - Select the GCM initialization phase, by writing GCMPH[1:0] with 0x0.
3. Write the initialization vector in AES_IVRx registers according to *Table 187*.
4. Write the key into the AES_KEYRx registers.
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN to start the calculation of the hash key. EN is automatically cleared when the calculation is completed.
7. Wait until the CCF flag is set in the AES_ISR register, indicating that the GCM hash subkey (H) computation is completed.
8. Clear the CCF flag by setting the CCF bit of the AES_ICR register.

GCM header phase

9. Initialize header phase:
   a) Select the GCM header phase, by writing 0x1 to GCMPH[1:0]. Do not change the other configurations written during GCM initialization.
   b) Enable the AES peripheral, by setting EN.
10. Append header data:
    a) If it is the last block and the AAD in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 24.4.5*.
    c) Repeat the step *b)* until the second-last AAD data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no AAD, that is, Len(A) = 0.

No data are read during header phase.

GCM payload phase

11. Initialize payload phase:
    a) Select the GCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
    b) If the header phase is skipped, enable the AES peripheral by setting EN.

12. Append payload data:
    a) If it is the last block and the message in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 24.4.5*, then read the AES_DOUTR register four times to save the resulting block
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(C)=0 (see GMAC mode).

GCM finalization

13. Select the GCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during GCM initialization.
14. Write the final GCM block into the AES_DINR register. It is the concatenated AAD bit and payload bit lengths, as shown in *Table 188*.
15. Wait until the CCF flag in the AES_ISR register is set.
16. Get the GCM authentication tag, by reading the AES_DOUTR register four times.
17. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
18. Disable the AES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data are written to AES_DINR normally (no swapping), while swapping is applied to tag data read from AES_DOUTR.

When transiting from the header or the payload phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.

Suspend/resume operations in GCM mode

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

To suspend the processing of a message, proceed as follows:

1. Wait until the CCF flag in the AES_ISR register is set (computation completed).
2. In the payload phase, read four times the AES_DOUTR register to save the last-processed block.
3. Clear the CCF flag of the AES_ISR register, by setting the CCF bit of the AES_ICR register. When GCM encryption payload phase is selected, verify that BUSY is cleared in AES_SR, to ensure that GF2mul hash function is completed.
4. Save the AES_SUSPRx registers in the memory.
5. In the payload phase, save the AES_IVRx registers as, during the data processing, they changed from their initial values. In the header phase, this step is not required.
6. Disable the AES peripheral, by clearing EN.
7. Save the current AES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

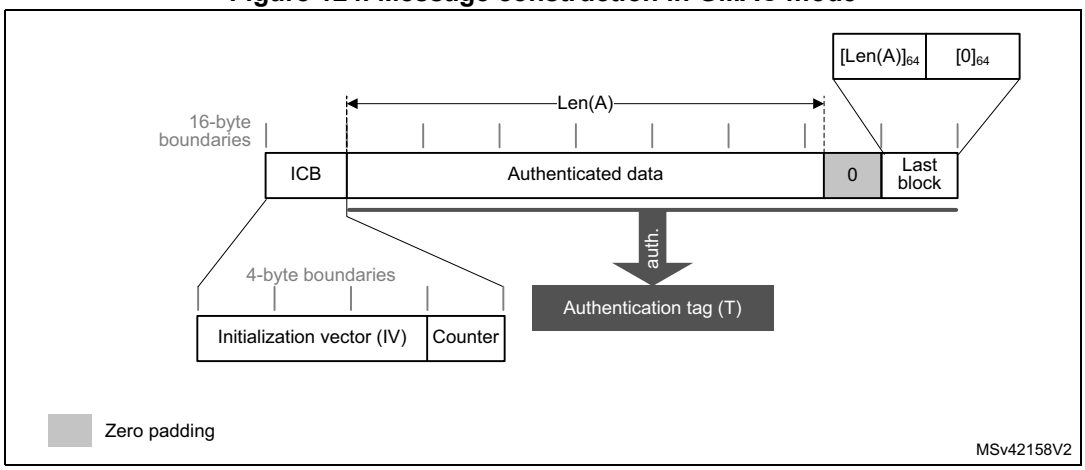
1. Disable the AES peripheral, by clearing EN.
2. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPRx registers.
3. In the payload phase, write the initialization vector register values, previously saved in the memory, back into their corresponding AES_IVRx registers. In the header phase, write initial setting values back into the AES_IVRx registers.
4. Restore the initial setting values in the AES_CR and AES_KEYRx registers.
5. Enable the AES peripheral, by setting EN.

24.4.12 AES Galois message authentication code (GMAC)

The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code).

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload). The following figure shows typical message construction for GMAC.

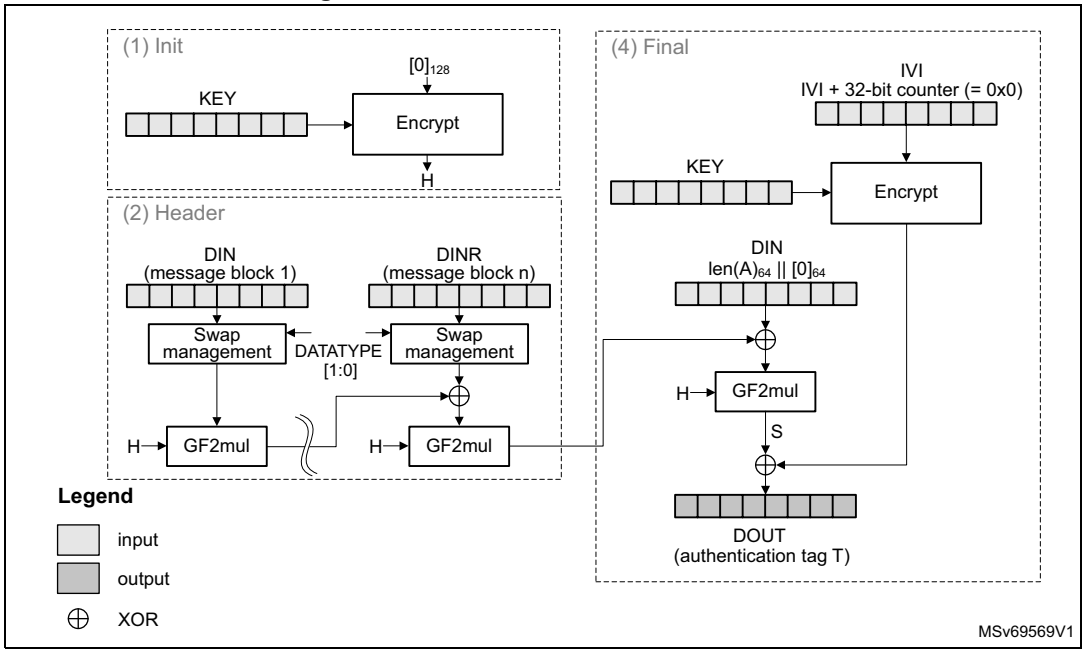
Figure 124. Message construction in GMAC mode



For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 125 describes the GMAC chaining implementation in the AES peripheral. To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 125. GMAC authentication mode



The GMAC algorithm corresponds to the GCM algorithm applied on a message that only contains a header. As a consequence, all steps and settings are the same as with the GCM, except that the payload phase is omitted.

Suspend/resume operations in GMAC

In GMAC mode, the sequence described for the GCM applies except that only the header phase can be interrupted.

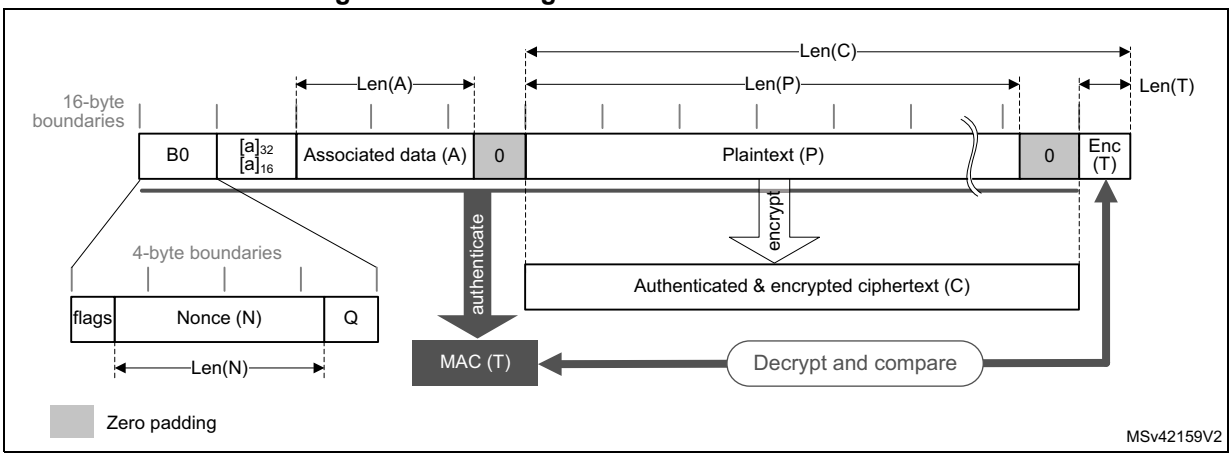
24.4.13 AES counter with CBC-MAC (CCM)

The AES counter with cipher block chaining-message authentication code (CCM) algorithm allows encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES counter mode processing. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve CBC-MAC as an authentication mode outside the context of the CCM specification.

The following figure shows typical message construction for CCM.

Figure 126. Message construction in CCM mode



The structure of the message is:

- **16-byte first authentication block (B0)**, composed of three distinct fields:
  - **Q:** a bit string representation of the octet length of P (Len(P))
  - **Nonce (N):** a single-use value (that is, a new nonce must be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags:** most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values **t** (MAC length expressed in bytes) and **Q** (plaintext length such that Len(P) < 2^{8Q} bytes). The counter blocks range associated to **Q** is equal to 2^{8Q-4}, that is, if the maximum value of **Q** is 8, the counter blocks used in cipher must be on 60 bits.
- **16-byte blocks (B)** associated to the associated data (A).
  This part of the message is only authenticated, not encrypted. This section has a known length Len(A) that can be a non-multiple of 16 bytes (see *Figure 126*). The standard also states that, on MSB bits of the first message block (B1), the associated data length expressed in bytes (a) must be encoded as follows:
  - If $0 < a < 2^{16} - 2^8$, then it is encoded as $[a]_{16}$, that is, on two bytes.
  - If $2^{16} - 2^8 < a < 2^{32}$, then it is encoded as 0xff || 0xfe || $[a]_{32}$, that is, on six bytes.
  - If $2^{32} < a < 2^{64}$, then it is encoded as 0xff || 0xff || $[a]_{64}$, that is, on ten bytes.
- **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length Len(P). This length can be a non-multiple of 16 bytes (see *Figure 126*).
- **Encrypted MAC (T)** of length Len(T) appended to the ciphertext C of overall length Len(C).

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

Note: CCM chaining mode can also be used with associated data only (that is, no payload).

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

N: 10111213 141516 (Len(N) = 56 bits or 7 bytes)

A: 00010203 04050607 (Len(A) = 64 bits or 8 bytes)

P: 20212223 (Len(P) = 32 bits or 4 bytes)

T: 6084341B (Len(T) = 32 bits or t = 4)

B0: 4F101112 13141516 00000000 00000004
B1: 00080001 02030405 06070000 00000000
B2: 20212223 00000000 00000000 00000000

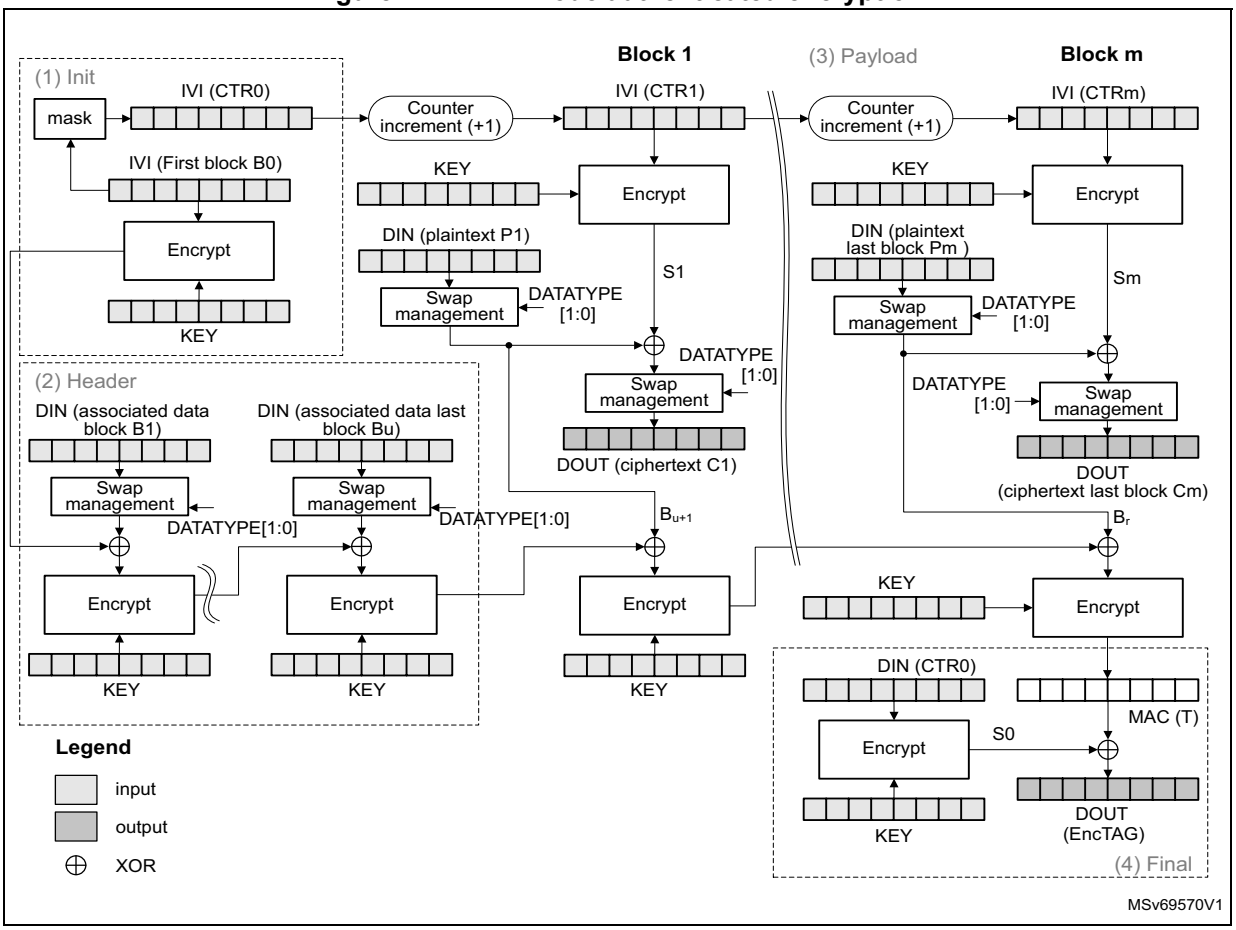
CTR0: 0710111213 141516 00000000 00000000

CTR1: 0710111213 141516 00000000 00000001

For more details, refer to NIST Special Publication 800-38C, *Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

Figure 127 describes the CCM chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x4.

Figure 127. CCM mode authenticated encryption



The first block of a CCM message (B0) must be prepared by the application as defined in *Table 189*.

Table 189. Initialization of IV registers in CCM mode

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
B0[127:96](1)	B0[95:64]	B0[63:32]	B0[31:0](2)

1. The five most significant bits are cleared (flag bits).
2. Q length bits are cleared, except for the bit 0 that is set.

AES supports counters up to 64 bits, as specified by NIST.

CCM encryption and decryption process

This process is described in *Section 24.4.6*, with the following sequence of events:

CCM initialize

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select CCM chaining mode (write CHMOD[2:0] with 0x4) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE.
   - Select the CCM initialization phase, by writing GCMPH[1:0] with 0x0.
3. Write the B0 data in AES_IVRx registers according to *Table 189*.
4. Write the key into the AES_KEYRx registers.
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN to start the first mask calculation. The EN bit is automatically cleared when the calculation is completed.
7. Wait until the CCF flag in the AES_ISR register is set.
8. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.

CCM header phase

9. Initialize header phase:
   a) Prepare the first block of the (B1) data associated with the message, in accordance with CCM chaining rules.
   b) Select the CCM header phase, by writing GCMPH[1:0] with 0x1. Do not change the other configurations written during the CCM initialization.
   c) Enable the AES peripheral, by setting EN.
10. Append header data:
    a) If it is the last block and the associated data in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 24.4.5*.
    c) Repeat the step *b)* until the second-last associated data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no associated data, that is, Len(A) = 0

No data are read during the header phase.

CCM payload phase

11. Initialize payload phase:
    a) Select the CCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during the CCM initialization.
    b) If the header phase is skipped, enable the AES peripheral, by setting EN.
12. Append payload data:
    a) In encryption only, if it is the last block and the plaintext in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 24.4.5*, then read the AES_DOUTR register four times to save the resulting block.
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(P) = 0 or Len(C) = Len(T).

Remove $LSB_{Len(T)}(C)$ encrypted tag information when decrypting ciphertext C.

CCM finalization

13. Select the CCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during the CCM initialization.
14. Wait until CCF flag in the AES_ISR register is set.
15. Get the CCM authentication tag, by reading the AES_DOUTR register four times.
16. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
17. Disable the AES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message. Mask the authentication tag output with tag length to obtain a valid tag.

Note: In the final phase, swapping is applied to tag data read from AES_DOUTR register.

When transiting from the header or the payload phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.

Suspend and resume operations in CCM mode

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

To suspend the processing of a message in header or payload phase, proceed as follows:

1. Wait until the CCF flag of the AES_ISR register is set (computation completed).
2. In the payload phase, read four times the AES_DOUTR register to save the last-processed block.
3. Clear the CCF flag in the AES_ISR register, by setting the CCF bit of the AES_ICR register.
4. Save the AES_SUSPRx registers in the memory.
5. Save the IV registers as they are altered during the data processing.
6. Disable the AES peripheral, by clearing EN.
7. Save the current AES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

1. Disable the AES peripheral, by clearing EN.
2. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPRx registers.
3. Restore AES_IVRx registers using the saved configuration.
4. Restore the initial setting values in the AES_CR and AES_KEYRx registers.
5. Enable the AES peripheral, by setting EN.

24.4.14 AES data registers and data swapping

Data input and output

A 16-byte data block enters the AES peripheral with four successive 32-bit word writes into the AES_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 16-byte data block is retrieved from the AES peripheral with four successive 32-bit word reads of the AES_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The four 32-bit words of a 16-byte data block must be stored in the memory consecutively and in big-endian order, that is, with the most significant word on the lowest address. See *Table 190* “no swapping” option for details.

Data swapping

The AES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the AES_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the AES_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through DATATYPE[1:0]. The selection applies to both AES input and output.

Note: The data in AES key registers (AES_KEYRx) and initialization vector registers (AES_IVRx) are not sensitive to the swap mode selection.

The AES data swapping feature is summarized in *Table 190* and *Figure 128*.

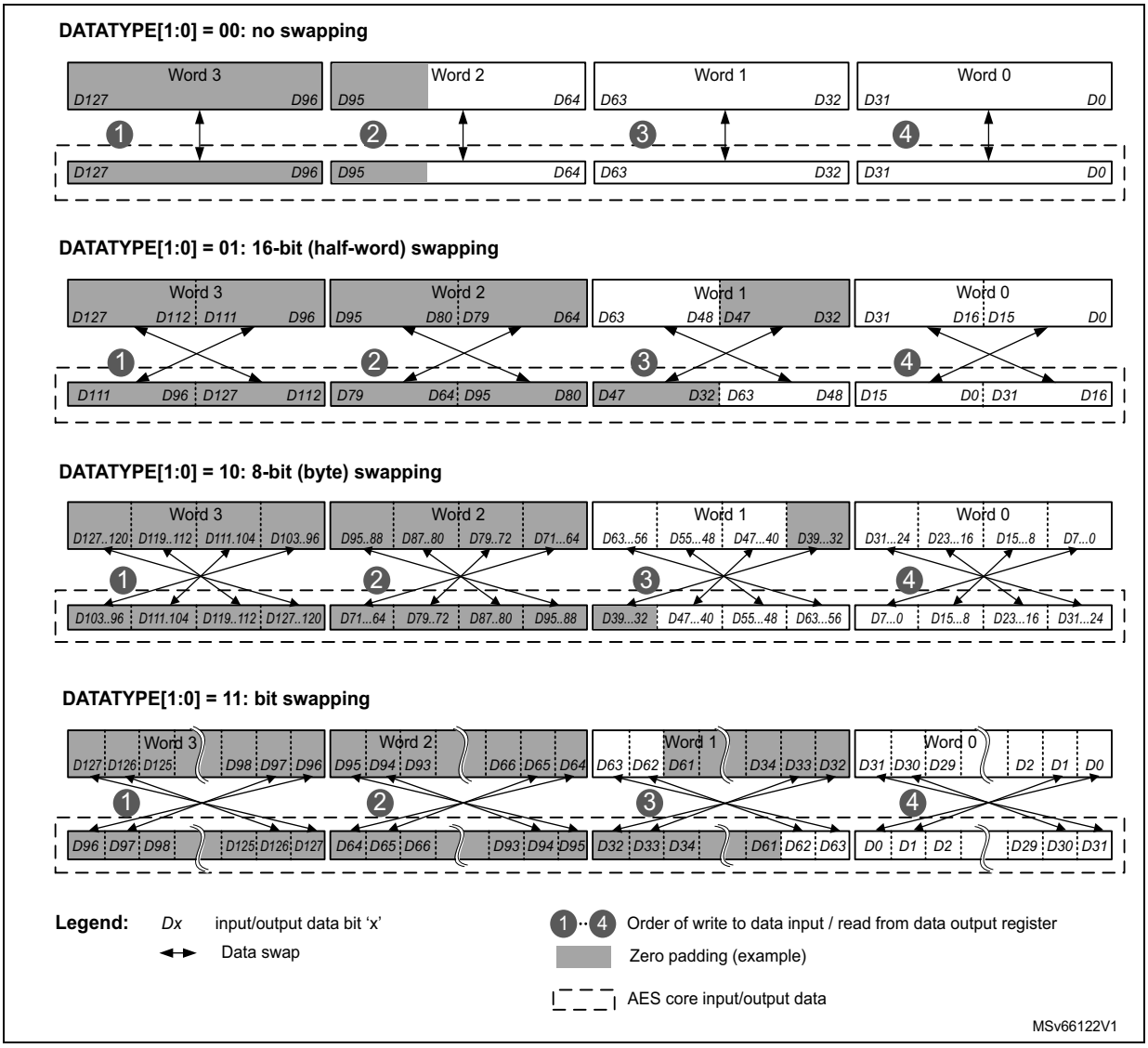
Table 190. AES data swapping example

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x0	No swapping	Block[127..64]: 0x04EEF672 2E04CE96 Block[63..0]: 0x4E6F7720 69732074
		Address @, word[127..96]: 0x04EEF672 Address @ + 0x4, word[95..64]: 0x2E04CE96 Address @ + 0x8, word[63..32]: 0x4E6F7720 Address @ + 0xC, word[31..0]: 0x69732074

Table 190. AES data swapping example (continued)

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x1	Half-word (16-bit) swapping	Block[63..0]: 0x**4E6F** 7720 **6973** 2074
		Address @, word[63..32]: 0x7720 **4E6F** Address @ + 0x4, word[31..0]: 0x2074 **6973**
0x2	Byte (8-bit) swapping	Block[63..0]: 0x**4E** 6F **77** 20 **69** 73 **20** 74
		Address @, word[63..32]: 0x20**77** 6F**4E** Address @ + 0x4, word[31..0]: 0x74**20** 73**69**
0x3	Bit swapping	Block[63..32]: 0x4E6F7720 0100 1110 0110 1111 0111 0111 0010 0000 Block[31..0]: 0x69732074 0110 1001 0111 0011 0010 0000 0111 0100
		Address @, word[63..32]: 0x04EE F672 0000 0100 1110 1110 1111 0110 0111 0010 Address @ + 0x4, word[31..0]: 0x2E04 CE96 0010 1110 0000 0100 1100 1110 1001 0110

Figure 128. 128-bit block construction according to the data type



Data padding

Figure 128 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 84 message bits, with DATATYPE[1:0] = 0x0
- 48 message bits, with DATATYPE[1:0] = 0x1
- 56 message bits, with DATATYPE[1:0] = 0x2
- 34 message bits, with DATATYPE[1:0] = 0x3

24.4.15 AES key registers

The eight AES_KEYRx write-only registers store the encryption or decryption key information, as shown on *Table 191*. Reads are not allowed for security reason.

Note: In memory and in AES *key registers, keys are stored in little-endian format, with most significant byte on the highest address.*

Table 191. Key endianness in AES_KEYRx registers (128/256-bit keys)

AES_KEYR7 [31:0]	AES_KEYR6 [31:0]	AES_KEYR5 [31:0]	AES_KEYR4 [31:0]	AES_KEYR3 [31:0]	AES_KEYR2 [31:0]	AES_KEYR1 [31:0]	AES_KEYR0 [31:0]
-	-	-	-	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
KEY[255:224]	KEY[223:192]	KEY[191:160]	KEY[159:128]	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]

The key registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield.

Write operations to the AES_KEYRx registers are ignored when AES peripheral is enabled (EN bit set). The application must check this before modifying key registers.

The entire key must be written before starting an AES computation.

The key registers must always be written in either ascending or descending order. The write sequence becomes:

- AES_KEYRx (x = 0 to 3 or x=3 to 0) for KEYSIZE cleared
- AES_KEYRx (x = 0 to 7 or x=7 to 0) for KEYSIZE set

Note: KEYSIZE must be written before the key.

As soon as the first key register is written, the KEYVALID flag is cleared. Once the key registers writing sequence is completed, KEYVALID is set and EN becomes writable. If an error occurs, KEYVALID is cleared and KEIF set (see *Section 24.4.17*).

24.4.16 AES initialization vector registers

The four AES_IVRx registers store the initialization vector (IV) information, as shown in *Table 192*. They can only be written if the AES peripheral is disabled (EN cleared).

Note: In memory and in AES IV registers, initialization vectors are stored in little-endian format, with most significant byte on the highest address.

Table 192. IVI bitfield spread over AES_IVRx registers

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0]

Initialization vector information depends on the chaining mode selected. When used, AES_IVRx registers are updated upon each AES computation cycle (useful for managing suspend mode).

The initialization vector registers are not affected by the data swapping feature controlled through DATATYPE[1:0].

24.4.17 AES error management

The AES peripheral manages the errors described in this section.

Read error flag (RDERRF)

Unexpected read attempt of the AES_DOUTR register returns zero, setting the RDERRF flag and the RWEIF flag. RDERRF is triggered during the computation phase or during the input phase.

Note: AES is not disabled when RDERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 24.6: AES interrupts*.

The RDERRF and RWEIF flags are cleared by setting the RWEIF bit of the AES_ICR register.

Write error flag (WRERRF)

Unexpected write attempt of the AES_DINR register is ignored, setting the WRERRF and the RWEIF flags. WRERRF is triggered during the computation phase or during the output phase.

Note: AES is not disabled when WRERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 24.6: AES interrupts*.

The WRERRF and RWEIF flags are cleared by setting the RWEIF bit of the AES_ICR register.

Key error interrupt flag (KEIF)

There are multiple sources of errors that set the KEIF flag of the AES_ISR register and clear the KEYVALID bit of the AES_SR register:

- **Key writing sequence error:** triggered upon detecting an incorrect sequence of writing key registers. See *Section 24.4.15: AES key registers* for details.

The KEIF flag is cleared with corresponding bit of the AES_ICR register. An interrupt is generated if the KEIE bit of the AES_IER register is set. For more details, refer to *Section 24.6: AES interrupts*.

Note: For any key error, clear KEIF flag prior to disabling and re-configuring AES.

24.5 AES in low-power modes

Table 193. Effect of low-power modes on AES

Mode	Description
Sleep	No effect.
Stop 0 and Stop 1	The AES register contents are kept.
Stop 2	The AES peripheral is powered down. It must be reinitialized upon existing this low-power mode.
Standby	

24.6 AES interrupts

There are multiple individual maskable interrupt sources generated by the AES peripheral to signal the following events:

- computation completed (CCF)
- read error (RDERRF)
- write error (WRERRF)
- key error (KEIF)

See *Section 24.4.17: AES error management* for details on AES errors.

These sources are combined into a common interrupt signal from the AES peripheral that connects to the Cortex® CPU interrupt controller. Application can enable or disable AES interrupt sources individually by setting/clearing the corresponding enable bit of the AES_IER register.

The status of the individual maskable interrupt sources can be read from the AES_ISR register. They are cleared by setting the corresponding bit of the AES_ICR register.

Table 194 gives a summary of the available features.

Table 194. AES interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable bit	Interrupt clear method
AES	computation completed flag	CCF	CCFIE	set CCF(1)
	read error flag	RDERRF(2)	RWEIE	set RWEIF(1)
	write error flag	WRERRF(2)		
	key error flag	KEIF	KEIE	set KEIF(1)

1. Bit of the AES_ICR register.
2. Flag of the AES_SR register, mirrored by the flag RWEIF of the AES_ISR register.

24.7 AES DMA requests

The AES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the DMAINEN and DMAOUTEN bits of the AES_CR register. When key derivation is selected (MODE[1:0] is at 0x1), setting those bits has no effect.

AES only supports single DMA requests.

Suspend and resume operations are not supported in DMA mode.

Detailed usage of DMA with AES can be found in *Appending data using DMA* subsection of *Section 24.4.5: AES encryption or decryption typical usage*.

Data input using DMA

Setting DMAINEN enables DMA writing into AES. AES then initiates, during the input phase, a set of single DMA requests for each 16-byte data block to write to the AES_DINR register (quadruple 32-bit word, MSB first).

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required (see Section 24.4.7).

Data output using DMA

Setting DMAOUTEN enables DMA reading from AES. AES then initiates, during the output phase, a set of single DMA requests for each 16-byte data block to read from the AES_DOUTR register (quadruple 32-bit word, MSB first).

After the output phase, at the end of processing of a 16-byte data block, AES switches automatically to a new input phase for the next data block, if any.

In DMA mode, the CCF flag has no use because the reading of the AES_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to managing the data transfers by software.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Stopping DMA transfers

All DMA request signals are de-asserted when AES is disabled (EN cleared) or the DMA enable bit (DMAINEN for input data, DMAOUTEN for output data) is cleared.

24.8 AES processing latency

The following tables provide the 16-byte data block processing latency per operating mode.

Table 195. Processing latency for ECB, CBC and CTR

Key size	Mode of operation	Chaining algorithm	Clock cycles
128-bit	Encryption or decryption(1)	ECB, CBC, CTR	51
	Key preparation	-	59
256-bit	Encryption or decryption(1)	ECB, CBC, CTR	75
	Key preparation	-	82

1. Excluding key preparation time (ECB and CBC only).

Table 196. Processing latency for GCM and CCM (in clock cycles)

Key size	Mode of operation	Chaining algorithm	Initialization phase	Header phase(1)	Payload phase(1)	Final phase(1)
128-bit	Encryption/ Decryption	GCM	64	35	51	59
		CCM	63	55	114	58
256-bit	Encryption/ Decryption	GCM	88	35	75	75
		CCM	87	79	162	82

1. Data insertion can include wait states forced by AES on the AHB bus (maximum 3 cycles, typical 1 cycle).

24.9 AES registers

The registers are accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

24.9.1 AES control register (AES_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]
rw								rw	rw	rw	rw		rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw

Bit 31 **IPRST**: AES peripheral software reset

Setting the bit resets the AES peripheral, putting all registers to their default values, except the IPRST bit itself. Hence, any key-relative data are lost. For this reason, it is recommended to set the bit before handing over the AES to a less secure application.

The bit must be kept low while writing any configuration registers.

Bits 30:24 Reserved, must be kept at reset value.

Bits 23:20 **NPBLB[3:0]:** Number of padding bytes in last block

This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.

0x0: All bytes are valid (no padding)

0x1: Padding for the last LSB byte

...

0xF: Padding for the 15 LSB bytes of last block.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **KEYSIZE:** Key size selection

This bitfield defines the key length in bits of the key used by AES.

0: 128-bit

1: 256-bit

Attempts to write the bit are ignored when the EN is set before the write access and it is not cleared by that write access.

Bit 17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **GCMPH[1:0]:** GCM or CCM phase selection
This bitfield selects the phase, applicable only with GCM, GMAC or CCM chaining modes.
0x0: Initialization phase
0x1: Header phase
0x2: Payload phase
0x3: Final phase

Bit 12 **DMAOUTEN**: DMA output enable
This bit enables automatic generation of DMA requests during the data phase, for outgoing data transfers from AES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bit 11 **DMAINEN**: DMA input enable
This bit enables automatic generation of DMA requests during the data phase, for incoming data transfers to AES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bits 10:7 Reserved, must be kept at reset value.

Bits 16, 6:5 **CHMOD[2:0]**: Chaining mode
This bitfield selects the AES chaining mode:
0x0: Electronic codebook (ECB)
0x1: Cipher-block chaining (CBC)
0x2: Counter mode (CTR)
0x3: Galois counter mode (GCM) and Galois message authentication code (GMAC)
0x4: Counter with CBC-MAC (CCM)
others: Reserved
Attempts to write the bitfield are ignored when EN is set before the write access and it is not cleared by that write access.

Bits 4:3 **MODE[1:0]**: Operating mode
This bitfield selects the AES operating mode:
0x0: Encryption
0x1: Key derivation (or key preparation), for ECB/CBC decryption only
0x2: Decryption
0x3: Reserved
Attempts to write the bitfield are ignored when EN is set before the write access and it is not cleared by that write access.

Bits 2:1 **DATATYPE[1:0]**: Data type
This bitfield defines the format of data written in the AES_DINR register or read from the AES_DOUTR register, through selecting the mode of data swapping. This swapping is defined in *Section 24.4.14: AES data registers and data swapping*.
0x0: No swapping (32-bit data).
0x1: Half-word swapping (16-bit data)
0x2: Byte swapping (8-bit data)
0x3: Bit-level swapping
Attempts to write the bitfield are ignored when EN is set before the write access and it is not cleared by that write access.

Bit 0 **EN**: Enable

This bit enables/disables the AES peripheral.

0: Disable

1: Enable

At any moment, clearing then setting the bit re-initializes the AES peripheral.

This bit is automatically cleared by hardware upon the completion of the key preparation (MODE[1:0] at 0x1) and upon the completion of GCM/GMAC/CCM initialization phase.

– The bit cannot be set as long as KEYVALID is cleared

24.9.2 AES status register (AES_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
								r				r	r	r	

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **KEYVALID**: Key valid flag

This bit is set by hardware when the key of size defined by KEYSIZE is loaded in AES_KEYRx key registers.

0: Key not valid

1: Key valid

The EN bit can only be set when KEYVALID is set.

The key must be written in the key registers in the correct sequence, otherwise the KEIF flag is set and KEYVALID remains cleared.

If set, KEIF must be cleared through the AES_ICR register, otherwise KEYVALID cannot be set. See the KEIF flag description for more details.

For further information on key loading, refer to *Section 24.4.15: AES key registers*.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy

This flag indicates whether AES is idle or busy.

0: Idle

1: Busy

AES is flagged as idle when disabled (when EN is low) or when the last processing is completed.

AES is flagged as busy when processing a block data, preparing a key (ECB or CBC decryption only).

When GCM encryption payload phase is selected, this flag must be at zero before suspending current process to manage a higher-priority message.

Bit 2 **WRERRF**: Write error flag

This bit is set when an unexpected write to the AES_DINR register occurred. When set WRERRF bit has no impact on the AES operations.

0: No error

1: Unexpected write to AES_DINR register occurred during computation or data output phase.

The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.

The flag is cleared by setting the RWEIF bit of the AES_ICR register.

Bit 1 **RDERRF**: Read error flag

This bit is set when an unexpected read to the AES_DOUTR register occurred. When set RDERRF bit has no impact on the AES operations.

0: No error

1: Unexpected read to AES_DOUTR register occurred during computation or data input phase.

The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.

The flag is cleared by setting the RWEIF bit of the AES_ICR register.

Bit 0 Reserved, must be kept at reset value.

24.9.3 AES data input register (AES_DINR)

Address offset: 0x008

Reset value: 0x0000 0000

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

Bits 31:0 **DIN[31:0]**: Data input

A four-fold sequential write to this bitfield during the Input phase results in writing a complete 16-bytes block of input data to the AES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 16-bytes input buffer.

Reads return zero.

24.9.4 AES data output register (AES_DOUTR)

Address offset: 0x00C

Reset value: 0x0000 0000

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

Bits 31:0 **DOUT[31:0]**: Data output

This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF flag set), virtually reads a complete 16-byte block of output data from the AES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield.

Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [31:0].

24.9.5 AES key register 0 (AES_KEYR0)

Address offset: 0x010

Reset value: 0x0000 0000

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

Bits 31:0 **KEY[31:0]**: Cryptographic key, bits [31:0]

These are bits [31:0] of the write-only bitfield KEY[255:0] AES encryption or decryption key, depending on the MODE[1:0] bitfield of the AES_CR register.

Writes to AES_KEYRx registers are ignored when AES is enabled (EN bit set).

A special writing sequence is required. In this sequence, any valid write to AES_KEYRx register clears the KEYVALID flag except for the sequence-completing write that sets it. Also refer to the description of the KEYVALID flag in the AES_SR register.

24.9.6 AES key register 1 (AES_KEYR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[63:32]**: Cryptographic key, bits [63:32]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

24.9.7 AES key register 2 (AES_KEYR2)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[95:64]**: Cryptographic key, bits [95:64]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

24.9.8 AES key register 3 (AES_KEYR3)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[127:96]**: Cryptographic key, bits [127:96]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

24.9.9 AES initialization vector register 0 (AES_IVR0)

Address offset: 0x020

Reset value: 0x0000 0000

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

Bits 31:0 **IVI[31:0]**: Initialization vector input, bits [31:0]

AES_IVRx registers store the 128-bit initialization vector or the nonce, depending on the chaining mode selected. This value is updated by hardware after each computation round (when applicable).
Write to this register is ignored when EN bit is set in AES_CR register

24.9.10 AES initialization vector register 1 (AES_IVR1)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[63:32]**: Initialization vector input, bits [63:32]

Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

24.9.11 AES initialization vector register 2 (AES_IVR2)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[95:80]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[95:64]**: Initialization vector input, bits [95:64]

Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

24.9.12 AES initialization vector register 3 (AES_IVR3)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[127:112]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[127:96]**: Initialization vector input, bits [127:96]
Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

24.9.13 AES key register 4 (AES_KEYR4)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[159:144]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[143:128]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[159:128]**: Cryptographic key, bits [159:128]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

24.9.14 AES key register 5 (AES_KEYR5)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[191:176]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[175:160]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[191:160]**: Cryptographic key, bits [191:160]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

24.9.15 AES key register 6 (AES_KEYR6)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[223:208]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[207:192]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[223:192]**: Cryptographic key, bits [223:192]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

24.9.16 AES key register 7 (AES_KEYR7)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[255:240]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[239:224]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[255:224]**: Cryptographic key, bits [255:224]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

24.9.17 AES suspend registers (AES_SUSPRx)

Address offset: 0x040 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

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

Bits 31:0 **SUSP[31:0]**: Suspend data

AES_SUSPRx registers contain the complete internal register states of the AES when the GCM, GMAC or CCM processing of the current task is suspended to process a higher-priority task. Refer to *Section 24.4.8: AES suspend and resume operations* for more details.

Clearing EN bit of the AES_CR register clears this register to zero.

AES_SUSPRx registers are not used in other chaining modes than GCM, GMAC or CCM.

24.9.18 AES interrupt enable register (AES_IER)

Address offset: 0x300

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIE	RWEIE	CCFIE
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIE:** Key error interrupt enable

This bit enables or disables (masks) the AES interrupt generation when KEIF (key error flag) is set.
0: Disabled (masked)
1: Enabled (not masked)

Bit 1 **RWEIE:** Read or write error interrupt enable

This bit enables or disables (masks) the AES interrupt generation when RWEIF (read and/or write error flag) is set.
0: Disabled (masked)
1: Enabled (not masked)

Bit 0 **CCFIE:** Computation complete flag interrupt enable

This bit enables or disables (masks) the AES interrupt generation when CCF (computation complete flag) is set.
0: Disabled (masked)
1: Enabled (not masked)

24.9.19 AES interrupt status register (AES_ISR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	RWEIF	CCF
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIF:** Key error interrupt flag

This read-only bit is set by hardware when the key information fails to load into key registers.
0: No key error detected
1: Key information failed to load into key registers
The flag setting generates an interrupt if the KEIE bit of the AES_IER register is set. It also clears the key registers and the KEYVALID flag in the AES_SR register.
The flag is cleared by setting the corresponding bit of the AES_ICR register.
KEIF is raised upon any of the following events:
–AES_KEYRx register write does not respect the correct order. (For KEYSIZE cleared, AES_KEYR0 then AES_KEYR1 then AES_KEYR2 then AES_KEYR3 register, or reverse. For KEYSIZE set, AES_KEYR0 then AES_KEYR1 then AES_KEYR2 then AES_KEYR3 then AES_KEYR4 then AES_KEYR5 then AES_KEYR6 then AES_KEYR7, or reverse).
KEIF must be cleared by the application software, otherwise KEYVALID cannot be set.

Bit 1 **RWEIF:** Read or write error interrupt flag

This read-only bit is set by hardware when a RDERRF or a WRERRF error flag is set in the AES_SR register.

0: No read or write error detected

1: Read or write error detected

The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.

The flag is cleared by setting the corresponding bit of the AES_ICR register.

The flags has no meaning when key derivation mode is selected.

See the AES_SR register for details.

Bit 0 **CCF:** Computation complete flag

This flag indicates whether the computation is completed. It is significant only when the DMAOUTEN bit is cleared, and it may stay high when DMAOUTEN is set.

0: Not completed

1: Completed

The flag setting generates an interrupt if the CCFIE bit of the AES_IER register is set.

The flag is cleared by setting the corresponding bit of the AES_ICR register.

24.9.20 AES interrupt clear register (AES_ICR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	RWEIF	CCF
													w	w	w

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIF:** Key error interrupt flag clear

Setting this bit clears the KEIF status bit of the AES_ISR register.

Bit 1 **RWEIF:** Read or write error interrupt flag clear

Setting this bit clears the RWEIF status bit of the AES_ISR register, and clears both RDERRF and WRERRF flags in the AES_SR register.

Bit 0 **CCF:** Computation complete flag clear

Setting this bit clears the CCF status bit of the AES_ISR register.

24.9.21 AES register map

Table 197. AES register map and reset values

Offset	Register name	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
0x000	AES_CR	IPRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]	Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	Reset value	0								0	0	0	0		0		0		0	0	0	0					0	0	0	0	0	0	0
0x004	AES_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
	Reset value																									0				0	0	0	
0x008	AES_DINR	DIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	AES_DOUTR	DOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	AES_KEYR0	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	AES_KEYR1	KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	AES_KEYR2	KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	AES_KEYR3	KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	AES_IVR0	IVI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	AES_IVR1	IVI[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	AES_IVR2	IVI[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	AES_IVR3	IVI[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	AES_KEYR4	KEY[159:128]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	AES_KEYR5	KEY[191:160]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	AES_KEYR6	KEY[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	AES_KEYR7	KEY[255:224]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 197. AES register map and reset values (continued)

Offset	Register name	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
0x040	AES_SUSPR0	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	AES_SUSPR1	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	AES_SUSPR2	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	AES_SUSPR3	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	AES_SUSPR4	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	AES_SUSPR5	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	AES_SUSPR6	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	AES_SUSPR7	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060-0x2FF	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x300	AES_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIE	RWEIE	CCFIE
	Reset value																														0	0	0
0x304	AES_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	RWEIF	CCF
	Reset value																														0	0	0
0x308	AES_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	RWEIF	CCF
	Reset value																														0	0	0
0x309-0x3FF	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

25 Hash processor (HASH)

25.1 HASH introduction

The hash processor is a fully compliant implementation of the secure hash algorithm (SHA-1, SHA-224, SHA-256) and the HMAC (keyed-hash message authentication code) algorithm. HMAC is suitable for applications requiring message authentication.

The hash processor computes FIPS (Federal Information Processing Standards) approved digests of length of 160, 224, 256 bits, for messages of any length less than 2^{64} bits.

25.2 HASH main features

- Suitable for data authentication applications, compliant with:
  - Federal Information Processing Standards Publication FIPS PUB 180-4, *Secure Hash Standard* (SHA-1 and SHA-2 family)
  - Federal Information Processing Standards Publication FIPS PUB 186-4, *Digital Signature Standard (DSS)*
  - Internet Engineering Task Force (IETF) Request For Comments RFC 2104, *HMAC: Keyed-Hashing for Message Authentication* and Federal Information Processing Standards Publication FIPS PUB 198-1, *The Keyed-Hash Message Authentication Code (HMAC)*
- Fast computation of SHA-1, SHA-224, SHA-256
  - 82 (respectively 66) clock cycles for processing one 512-bit block of data using SHA-1 (respectively SHA-256) algorithm
- Support for HMAC mode with all supported algorithm
- Corresponding 32-bit words of the digest from consecutive message blocks are added to each other to form the digest of the whole message
  - Automatic 32-bit words swapping to comply with the internal little-endian representation of the input bit-string
  - Supported word swapping format: bits, bytes, half-words, and 32-bit words
- Single 32-bit, write-only, input register associated to an internal input FIFO that can receive a single data block of size depending on the selected algorithm
- Automatic padding to complete the input bit string to fit digest minimum block size
- AMBA AHB target peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
- 8 × 32-bit words (H0 to H7) for output message digest
- Automatic data flow control supporting direct memory access (DMA) using one channel.
- Support for both single and fixed DMA burst transfers of four words.
- Interruptible message digest computation, on a per-block basis (512 bits)
  - Reloadable digest registers
  - Hashing computation suspend/resume mechanism, including DMA

25.3 HASH implementation

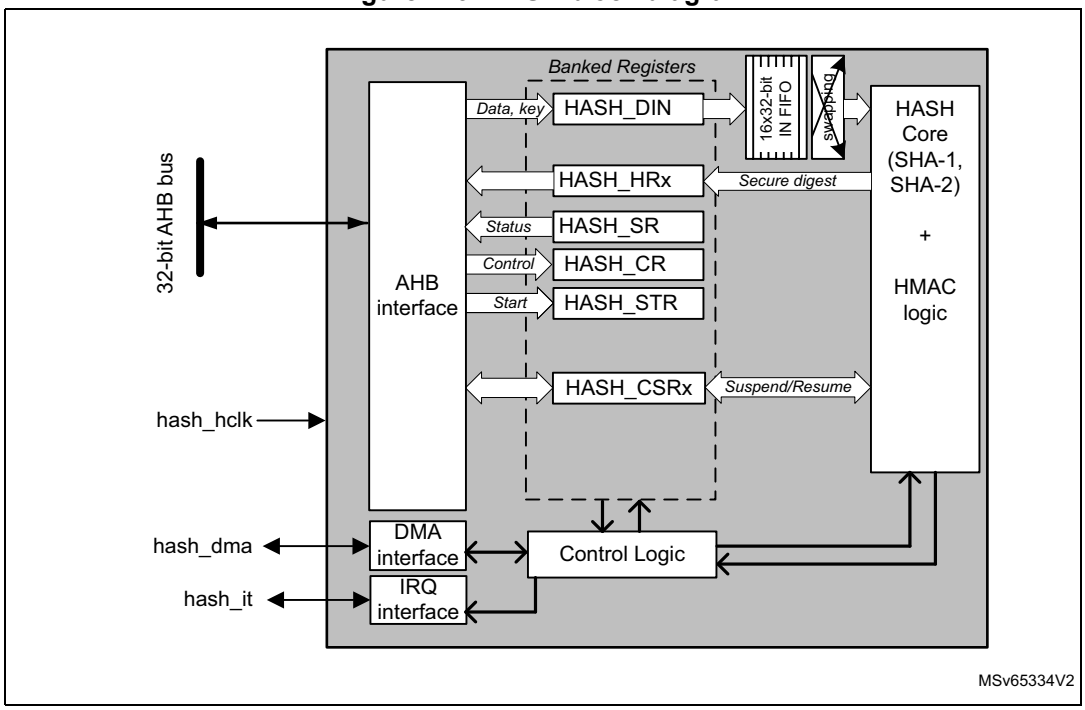
The devices have a single instance of HASH peripheral.

25.4 HASH functional description

25.4.1 HASH block diagram

Figure 129 shows the block diagram of the hash processor.

Figure 129. HASH block diagram



25.4.2 HASH internal signals

Table 198 describes a list of useful to know internal signals available at HASH level, not at product level (on pads).

Table 198. HASH internal input/output signals

Signal name	Signal type	Description
hash_hclk	digital input	AHB bus clock
hash_it	digital output	Hash processor global interrupt request
hash_dma	digital input/output	DMA burst request/ acknowledge

25.4.3 About secure hash algorithms

The hash processor is a fully compliant implementation of the secure hash algorithm defined by FIPS PUB 180-4 standard.

With each algorithm, the HASH computes a condensed representation of a message or data file. More specifically, when a message is presented on the input, the HASH processing core produces a fixed-length output string called a message digest (see *Table 199*).

Table 199. Information on supported hash algorithms

Algorithm	Message digest size (in bits)	Block size (in bytes)	Message length	Bit string message
SHA-1	160	64	$< 2^{64}$ bits	Yes
SHA-224	224			
SHA-256	256			

The message digest can then be processed with a digital signature algorithm in order to generate or verify the signature for the message.

Signing the message digest rather than the message often improves the efficiency of the process since the message digest is usually much smaller in size than the message. The verifier of a digital signature has to use the same hash algorithm as the one used by the creator of the digital signature.

The SHA-2 functions supported by the hash processor are qualified as “secure” by NIST because it is computationally infeasible to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest (SHA-1 does not qualify as secure since February 2017).

25.4.4 Message data feeding

The message (or data file) to be processed by the HASH must be considered as a bit string. Per FIPS PUB 180-4 standard this message bit string grows from left to right, with hexadecimal words expressed in “big-endian” convention, so that within each word, the most significant bit is stored in the left-most bit position. For example message string “abc” with a bit string representation of “**01100001 01100010 01100011**” is represented by a 32-bit word **0x00636261**, and 8-bit words **0x61626300**.

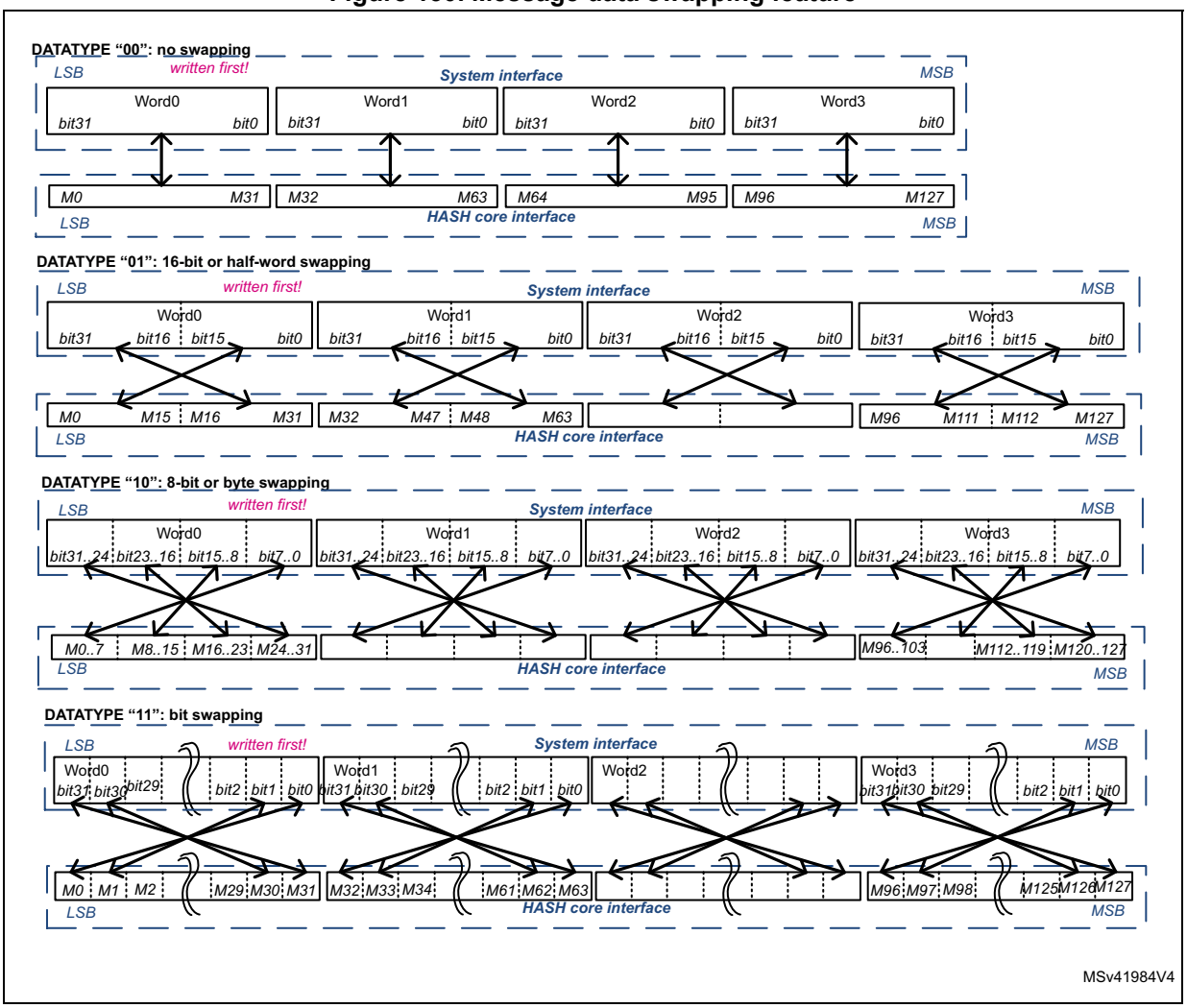
Data are entered into the HASH one 32-bit word at a time, by writing them into the HASH_DIN register. The current contents of the HASH_DIN register are transferred to the 16 words input FIFO each time the register is written with new data. Hence, HASH_DIN and the FIFO form a seventeen 32-bit words length FIFO (named the IN buffer).

In accordance to the kind of data to be processed (for example byte swapping when data are ASCII text stream) there must be a bit, byte, half-word, or no swapping operation to be performed on data from the input FIFO before entering the little-endian hash processing core. *Figure 130* shows how the hash processing core 32-bit data block M0...31 is constructed from one 32-bit words popped into input FIFO by the driver, according to the DATATYPE bitfield in the HASH control register (HASH_CR)

HASH_DIN data endianness when bit swapping is disabled (DATATYPE = 00) can be described as following: the least significant bit of the message has to be at MSB position in

the first word entered into the hash processor, the 32nd bit of the bit string has to be at MSB position in the second word entered into the hash processor and so on.

Figure 130. Message data swapping feature



25.4.5 Message digest computing

The hash processor sequentially processes several blocks when computing the message digest. Block sizes can be found in *Table 199: Information on supported hash algorithms*.

Each time the DMA or the CPU writes a block to the hash processor, the HASH automatically starts computing the message digest. This operation is known as partial digest computation.

As described in *Section 25.4.4: Message data feeding*, the message to be processed is entered into the HASH 32-bit word at a time, writing to the HASH_DIN register to fill the input FIFO. In order to perform the hash computation on this data the application must follow below sequence.

1. Initialize the hash processor using the HASH_CR register:
   – Select the right algorithm using the ALGO[1:0] field. If needed, program the correct swapping operation on the message input words using the DATATYPE[1:0] bitfield.
   – When HMAC mode is required, set the MODE bit as well as the LKEY bit if the HMAC key size is greater than the known block size of the algorithm (otherwise keep LKEY cleared). Refer to *Section 25.4.7: HMAC operation* for details.
   – Update NBLW[4:0] in the HASH_STR register to define the number of valid bits in the last word of the message if it is different from 32 bits. NBLW information is used to correctly perform the automatic message padding before the final message digest computation.
2. Complete the initialization by setting the INIT bit in HASH_CR register. Also set the DMAE bit if data are transferred via DMA.

Caution: When programming step 2, it is important that the correct configuration values (ALGO, DATATYPE, HMAC mode, key length, NBLW) are set up before or at the same time.

3. Start filling data by writing to the HASH_DIN register, unless data are automatically transferred via DMA. Note that the processing of a block can start only once the last value of the block has entered the input FIFO. The way the partial or final digest computation is managed depends on the way data are fed into the processor:
   – Data are filled by software:

     Partial digest computations are triggered each time the application writes the first word of the next block, the block size being defined by NBWE bits in HASH_SR. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write new data to HASH_DIN. This mechanism avoids the introduction of wait states by the HASH.

     The final digest computation is triggered when the last block is entered and the software sets the DCAL bit. If the message length is not an exact multiple of the block size, the NBLW field in HASH_STR register must be written prior to writing DCAL bit (see *Section 25.4.6* for details).
   – Data are filled as a single DMA transfer (MDMAT = 0):

     Partial digest computations are triggered automatically each time the FIFO is full.

     The final digest computation is triggered automatically when the last block has been transferred to the HASH_DIN register by DMA (DCAL bit is set by hardware). If the message length is not an exact multiple of the block size, the NBLW field in HASH_STR register must be written prior to enabling the DMA (see *Section 25.4.6* for details).
   – Data are filled using multiple DMA transfers (MDMAT = 1):

     Partial digest computations are triggered as for single DMA transfers (refer to the above description). However, the final digest computation is not triggered automatically when the last block has been transferred by DMA to the HASH_DIN register (DCAL bit is not set by hardware). It enables the hash processor to receive a new DMA transfer as part of this digest computation. To launch the final digest computation, the software must clear MDMAT bit before the last DMA transfer in order to trigger the final digest computation as it is done for single DMA transfers.
4. Once the digest calculation is completed (DCIS = 1), the resulting digest can be read from the output registers, as described in *Table 200*. When set, the DMAEN bit is automatically cleared by hardware.

Table 200. Hash processor outputs

Algorithm	Valid output registers	Most significant bit	Digest size (in bits)
SHA-1	HASH_H0 to HASH_H4	HASH_H0[31]	160
SHA-224	HASH_H0 to HASH_H6	HASH_H0[31]	224
SHA-256	HASH_H0 to HASH_H7		256

For more information about HMAC detailed instructions, refer to *Section 25.4.7: HMAC operation*.

Aborting HASH operation

HASH can be disabled at any time by writing 0x4 to HASH_CR, after stopping any running DMA transfer to HASH (if applicable).

25.4.6 Message padding

Overview

When computing a condensed representation of a message, the process of feeding data into the hash processor (with automatic partial digest computation every block size transfer) loops until the last bits of the original message are written to the HASH_DIN register.

As the length (number of bits) of a message can be any integer value, the last word written to the hash processor may have a valid number of bits between 1 and 32. This number of valid bits in the last word, NBLW, has to be written to the HASH_STR register, so that message padding is correctly performed before the final message digest computation.

Padding processing

Detailed hashing sequences with padding are described in *Section 25.4.5: Message digest computing*. As mentioned in these sequences, the padding is done automatically using the information provided in NBLW[4:0].

Hashing example

Below is an example of SHA-1 digest computation with HASH. Message processing and padding are defined in Federal Information Processing Standards PUB 180-4.

Let us assume that the original message is the ASCII binary-coded form of “abc”, of length L = 24 (‘**U**’ means don’t care):

```
byte 0    byte 1   byte 2   byte 3
01100001 01100010 01100011 UUUUUUUU
<-- 1st word written to HASH_DIN -->
```

NBLW has to be loaded with the value 24: a “**1**” is appended at bit location 24 in the bit string (starting counting from left to right in the above bit string), which corresponds to bit 31 in the HASH_DIN register (little-endian convention):

```
01100001 01100010 01100011 1UUUUUUU
```

Since L = 24, the number of bits in the above bit string is 25, and 423 “0” bits are automatically appended, making now 448 bits.

This gives in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000
```

The message length value, **L**, in two-word format (that is 00000000 00000018) is appended. Hence, the final padded message in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000018
```

If the hash processor is programmed to swap byte within HASH_DIN input register (DATATYPE = 10 in HASH_CR), the above message has to be entered using following sequence:

1. **0xUU636261** is written to the HASH_DIN register (where '**U**' means don't care)
2. **0x18** is written to the HASH_STR register (the number of valid bits in the last word written to the HASH_DIN register is 24, as the original message length is 24 bits)
3. **0x100** is written to the HASH_STR register to run the final digest computation.
4. The hash computing is complete with the message digest available in the HASH_HRx registers (x = 0 to 4) for the SHA-1 algorithm. For this FIPS example, the expected value is as follows:

```
HASH_HR0 = 0xA9993E36
HASH_HR1 = 0x4706816A
HASH_HR2 = 0xBA3E2571
HASH_HR3 = 0x7850C26C
HASH_HR4 = 0x9CD0D89D
```

25.4.7 HMAC operation

Overview

As specified by Internet Engineering Task Force RFC2104 and NIST FIPS PUB 198-1, the HMAC algorithm is used for message authentication by irreversibly binding the message being processed to a key chosen by the user. The algorithm consists of two nested hash operations:

```
HMAC(message)= Hash((Key | pad) XOR opad |
               Hash((Key | pad) XOR ipad | message))
```

Where:

- **opad** = $[0x5C]_n$ (outer pad) and **ipad** = $[0x36]_n$ (inner pad)
- $[X]_n$ represents a repetition of X *n* times, where *n* equal to the byte size of the underlying hash function data block (n = 64 when block size is 512 bits).
- **pad** is a sequence of zeroes needed to extend the key to the length *n* defined above. If the key length is greater than *n*, the application must first hash the key using Hash() function and then use the resultant byte string as the actual key to HMAC.
- | represents the concatenation operator.

Note: HMAC mode of the hash processor can be used with all supported algorithms.

HMAC processing

Four different steps are required to compute the HMAC:

1. The software sets the INIT bit, with the MODE bit set and the ALGO bits selecting the desired algorithm. The LKEY bit must also be set if the key being used is longer than 64 bytes. In this case, as required by HMAC specifications, the hash processor uses the hash of the key instead of the real key.
2. The software provides the key to be used for the inner hash function, using the same mechanism as the message string loading that is by writing the key data into HASH_DIN register and then completing the transfer by setting DCAL bit and the correct NBLW to HASH_STR register.
3. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write the message string to HASH_DIN. When the last word of the last block is entered and the software sets DCAL bit in HASH_STR register, the NBLW bitfield must be programmed at the same time to a value different from zero if the message length is not an exact multiple of the block size. Note that the DMA can also be used to feed the message string, as described in *Section 25.4.5: Message digest computing*.
4. Once the processor is ready again (DINIS = 1 in HASH_SR), the software provides the key to be used for the outer hash function, writing the key data into HASH_DIN register, and then completing the transfer by setting DCAL bit and programming the correct NBLW to HASH_STR register. The HMAC result can be found in the valid output registers (HASH_HRx) as soon as DCIS bit is set.

Note: The computation latency of the HMAC primitive depends on the lengths of the keys and message, as described in Section 25.4.11: HASH processing time.

Endianness management details can be found in Section 25.4.4: Message data feeding.

HMAC example

Below is an example of HMAC SHA-1 algorithm (ALGO = 00 and MODE = 1 in HASH_CR) as specified by NIST. SHA-1 block size is 64 bytes.

Let us assume that the original message is the ASCII binary-coded form of "**`Sample message for keylen = blocklen`**", of length L = 34 bytes. If the HASH is programmed in no swapping mode (DATATYPE = 00 in HASH_CR), the following data must be loaded sequentially into HASH_DIN register:

1. **Inner hash key** input (length = 64, that is, no padding), specified by NIST. As key length = 64, LKEY bit is cleared in HASH_CR register

   `00010203 04050607 08090A0B 0C0D0E0F 10111213 14151617`
   `18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F`
   `30313233 34353637 38393A3B 3C3D3E3F`

2. **Message** input (length = 34, that is, padding required). HASH_STR must be set to **`0x20`** to start message padding and inner hash computation (see **'U**' as don't care)

   `53616D70 6C65206D 65737361 67652066 6F72206B 65796C65`
   `6E3D626C 6F636B6C 656EUUUU`

3. **Outer hash key** input (length = 64, that is, no padding). A key identical to the inner hash key is entered here.

4. **Final outer hash computing** is then performed by the HASH. The HMAC-SHA1 digest result is available in the HASH_HRx registers, as shown below:

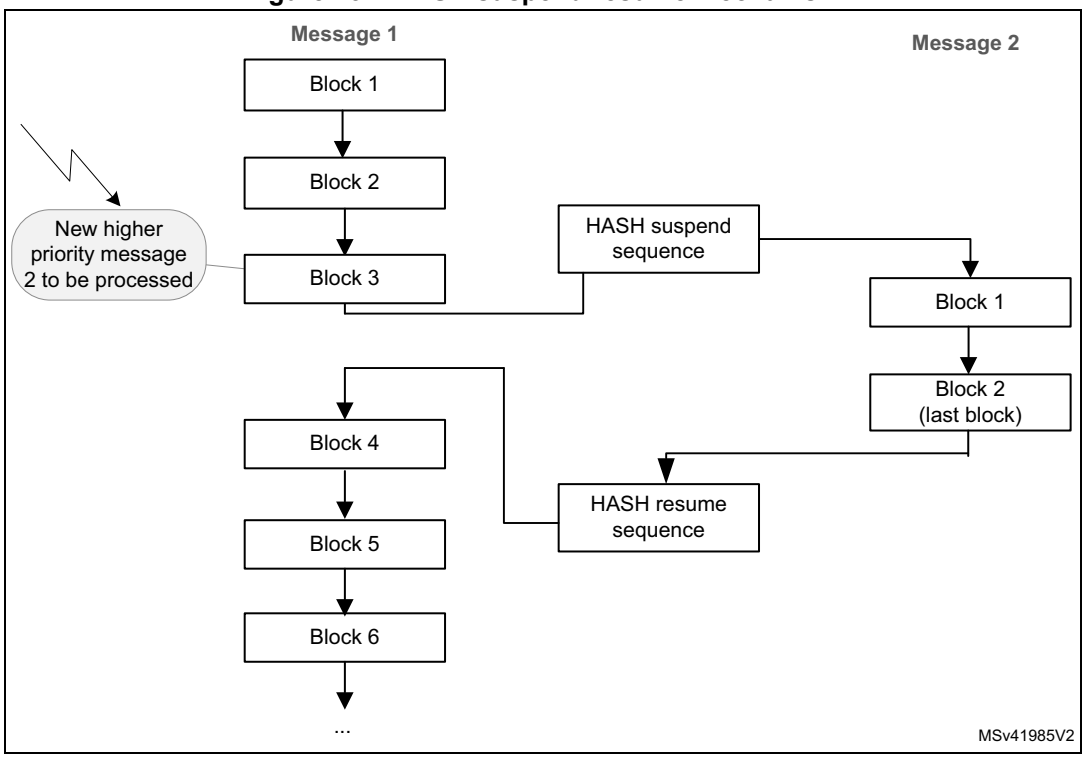
```
HASH_HR0 = 0x5FD596EE
HASH_HR1 = 0x78D5553C
HASH_HR2 = 0x8FF4E72D
HASH_HR3 = 0x266DFD19
HASH_HR4 = 0x2366DA29
```

25.4.8 HASH suspend/resume operations

Overview

It is possible to interrupt a hash/HMAC operation to perform another processing with a higher priority. The interrupted process completes later when the higher-priority task has been processed, as shown in *Figure 131*.

Figure 131. HASH suspend/resume mechanism



To do so, the context of the interrupted task must be saved from the HASH registers to memory, and then be restored from memory to the HASH registers.

The procedures where the data flow is controlled by software or by DMA are described hereafter.

Data loaded by software

When the DMA is not used to load the message into the hash processor, the context can be saved only when no block processing is ongoing.

To suspend the processing of a message, proceed as follows after writing the number of words defined in NBWE:

1. In Polling mode, wait for BUSY = 0 then poll if the DINIS status bit is set.
   In Interrupt mode, implement the next step in DINIS interrupt handler (recommended).
2. Store the contents of the following registers into memory:
   - HASH_IMR
   - HASH_STR
   - HASH_CR
   - HASH_CSR0 to HASH_CSR37, plus HASH_CSR38 to HASH_CSR53 if an HMAC operation was ongoing

To resume the processing of a message, proceed as follows:

1. Write the following registers with the values saved in memory: HASH_IMR, HASH_STR, HASH_CR.
2. Initialize the hash processor by setting the INIT bit in the HASH_CR register
3. Write the HASH_CSRx registers with the values saved in memory.
4. Restart the processing from the point where it has been interrupted.

Data loaded by DMA

When the DMA is used to load the message into the hash processor, it is recommended to suspend and then restore a secure digest computing as described below.

In this sequence the DMA channel allocated to the hash peripheral remains allocated to the processing of message 1 (see *Figure 131*).

To suspend the processing of a message using DMA, proceed as follows:

1. Clear the DMAE bit to disable the DMA interface. The hash peripheral automatically fetches enough data via the DMA to complete the current burst transfer.
2. Wait until the last DMA transfer is complete (DMAS = 0 in HASH_SR).
3. Disable the DMA channel.
4. In Polling or Interrupt mode (recommended), wait until the hash processor is ready (no block is being processed), that is wait for DINIS = 1 in HASH_SR. If DCIS is also set in HASH_SR, the hash result is available and the context swapping is useless. Else go to step 5.
5. Save HASH_IMR, HASH_STR, HASH_CR, and HASH_CSR0 to HASH_CSR37 registers. HASH_CSR38 to HASH_CSR53 registers must also be saved if an HMAC operation was ongoing.

To resume the processing of a message using DMA, proceed as follows:

1. Reconfigure the DMA controller so that it proceeds with the transfer of the message up to the end if it is not interrupted again.
2. Program the values saved in memory to HASH_IMR, HASH_STR and HASH_CR registers.
3. Initialize the hash processor by setting the INIT bit in the HASH_CR register.
4. Program the values saved in memory to the HASH_CSRx registers.
5. Restart the processing from the point where it was interrupted by setting the DMAE bit.

25.4.9 HASH DMA interface

The HASH supports both single and fixed DMA burst transfers of four words.

The hash processor provides an interface to connect to the DMA controller. This DMA can be used to write data to the HASH by setting the DMAE bit in the HASH_CR register. When this bit is set, the HASH initiates a DMA request each time a block has to be written to the HASH_DIN register.

Once four 32-bit words have been received, the HASH automatically triggers a new request to the DMA. For more information, refer to *Section 25.4.5: Message digest computing*.

Before starting the DMA transfer, the software must program the number of valid bits in the last word that is copied into HASH_DIN register. This is done by writing in HASH_STR register the following value:

`NBLW = Len(Message)% 32` where "**`x%32`**" gives the remainder of x divided by 32.

The DMAS bit of the HASH_SR register provides information on the DMA interface activity. This bit is set with DMAE and cleared when DMAE is cleared and no DMA transfer is ongoing.

Note: No interrupt is associated to DMAS bit.

When MDMAT is set, the size of the transfer must be a multiple of four words.

25.4.10 HASH error management

No error flags are generated by the hash processor.

25.4.11 HASH processing time

Table 201 summarizes the time required to process an intermediate block for each mode of operation.

Table 201. Processing time (in clock cycle)

Mode of operation	Block size (in bytes)	FIFO load(1)	Computation phase	Total
SHA-1	64	16	66	**82**
SHA-224		16	50	**66**
SHA-256				

1. Add the time required to load the block into the processor.

The time required to process the last block of a message (or of a key in HMAC) can be longer. This time depends on the length of the last block and the size of the key (in HMAC mode).

Compared to the processing of an intermediate block, it can be increased by the factor below:

- **1 to 2.5** for a hash message
- **~2.5** for an HMAC input-key
- **1 to 2.5** for an HMAC message
- **~2.5** for an HMAC output key in case of a short key
- **3.5 to 5** for an HMAC output key in case of a long key

25.5 HASH interrupts

Two individual maskable interrupt sources are generated by the hash processor to signal the following events:

- Digest calculation completion (DCIS)
- Data input buffer ready (DINIS)

Both interrupt sources are connected to the same global interrupt request signal (hash_it), which is in turn connected to the device interrupt controller. Each interrupt source can individually be enabled or disabled by changing the mask bits in the HASH_IMR register. Setting the appropriate mask bit enables the interrupt.

The status of each maskable interrupt source can be read from the HASH_SR register. *Table 202* gives a summary of the available features.

Table 202. HASH interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
HASH	Digest computation completed	DCIS	DCIE	Clear DCIS or set INIT
	Data input buffer ready to get a new block	DINIS	DINIE	Clear DINIS or write to HASH_DIN

25.6 HASH registers

The hash core is associated with several control and status registers and several message digest registers. All these registers are accessible through 32-bit word accesses only, else an AHB error is generated.

25.6.1 HASH control register (HASH_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[1:0]		LKEY
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MDMAT	DINNE	NBW[3:0]				Res.	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
		rw	r	r	r	r	r		rw	rw	rw	rw	w		

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:17 **ALGO[1:0]**: Algorithm selection

These bits select the hash algorithm. This selection is only taken into account when the INIT bit is set. Changing this bitfield during a computation has no effect

When the ALGO bitfield is updated and INIT bit is set, NBWE in HASH_SR is automatically updated to 0x11.

00: SHA-1

01: reserved

10: SHA-224

11: SHA-256

Bit 16 **LKEY**: Long key selection

The application must set this bit if the HMAC key is greater than the block size (64 bytes)

This selection is only taken into account when the INIT and MODE bits are set (HMAC mode selected). Changing this bit during a computation has no effect.

0: HMAC key is shorter or equal to the block size (short key). The actual key value written in HASH_DIN is used during the HMAC computation.

1: HMAC key is longer than the block size (long key). The hash of the key is used instead of the real key during the HMAC computation.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **MDMAT**: Multiple DMA transfers

This bit is set when hashing large files when multiple DMA transfers are needed.

0: DCAL is automatically set at the end of a DMA transfer.

1: DCAL is not automatically set at the end of a DMA transfer.

Bit 12 **DINNE**: DIN not empty

Refer to DINNE bit of HASH_SR for a description of DINNE bit.

This bit is read-only.

Bits 11:8 **NBW[3:0]:** Number of words already pushed

Refer to NBWP[3:0] bitfield of HASH_SR for a description of NBW[3:0] bitfield.

This bitfield is read-only.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **MODE:** Mode selection

This bit selects the normal or the keyed HMAC mode for the selected algorithm. This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

0: Hash mode selected

1: HMAC mode selected. LKEY bit must be set if the key being used is longer than 64 bytes.

Bits 5:4 **DATATYPE[1:0]:** Data type selection

This bitfield defines the format of the data entered into the HASH_DIN register:
00: 32-bit data. The data written into HASH_DIN are directly used by the HASH processing, without reordering.
01: 16-bit data or half-word. The data written into HASH_DIN are considered as two half-words, and are swapped before being used by the HASH processing.
10: 8-bit data or bytes. The data written into HASH_DIN are considered as four bytes, and are swapped before being used by the HASH processing.
11: bit data or bit string. The data written into HASH_DIN are considered as 32 bits (1st bit of the string at position 0), and are swapped before being used by the HASH processing (1st bit of the string at position 31).

Bit 3 **DMAE:** DMA enable

Setting this bit enables DMA transfers. After this bit is set, it is cleared by hardware while the last data of the message is written into the hash processor.
Clearing this bit while a DMA transfer is ongoing does not abort the current transfer. Instead, the DMA interface of the HASH remains internally enabled until the transfer is complete or INIT is set.
Setting INIT bit does not clear DMAE bit.
0: DMA transfers disabled
1: DMA transfers enabled. A DMA request is sent as soon as the hash core is ready to receive data.

Bit 2 **INIT:** Initialize message digest calculation

This is a write-only bit always returning zero upon reading. Setting this bit resets the hash processor core, so that HASH is ready to compute the message digest of a new message. Clearing this bit has no effect.

Bits 1:0 Reserved, must be kept at reset value.

25.6.2 HASH data input register (HASH_DIN)

Address offset: 0x004

Reset value: 0x0000 0000

HASH_DIN is the data input register. It is 32-bit wide. This register is used to enter the message by blocks of 64 bytes.

When the HASH_DIN register is programmed, the value presented on the AHB bus is ‘pushed’ into the hash core and the register takes the new value presented on the AHB bus. To get a correct message format, the DATATYPE bits must have been previously configured in the HASH_CR register.

When a complete block has been written to the HASH_DIN register, an intermediate digest calculation is launched:

- by writing first data of the next block into the HASH_DIN register, if the DMA is not used
- automatically, if the DMA is used

When the last block has been written to the HASH_DIN register, the final digest calculation (including padding) is launched by setting the DCAL bit in the HASH_STR register (final digest calculation). This operation is automatic if the DMA is used and MDMAT bit is cleared.

Reading the HASH_DIN register returns zeros.

Note: When the HASH is busy, a write access to the HASH_DIN register might stall the AHB bus if the digest calculation (intermediate or final) is not complete.

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

Bits 31:0 **DATAIN[31:0]:** Data input

Writing this register pushes the current register content into the FIFO, and the register takes the new value presented on the AHB bus.

Do not write to this register when BUSY is set.

Reading this register returns zeros.

25.6.3 HASH start register (HASH_STR)

Address offset: 0x008

Reset value: 0x0000 0000

The HASH_STR register has two functions:

- It is used to define the number of valid bits in the last word of the message entered in the hash processor (that is the number of valid least significant bits in the last data written to the HASH_DIN register)
- It is used to start the processing of the last block in the message by setting the DCAL bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
							w				rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **DCAL:** Digest calculation

This is a write-only bit always returning zero upon reading. Setting this bit starts the message padding using the previously written value of NBLW, and starts the calculation of the final message digest with all the data words written to the input FIFO since the INIT bit was last set.

Do not set DCAL when BUSY is set.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **NBLW[4:0]:** Number of valid bits in the last word

When the last word of the message bit string is written to HASH_DIN register, the hash processor takes only the valid bits, specified as below, after internal data swapping.

The above mechanism is valid only if DCAL = 0. If NBLW bits are written while DCAL is set, the NBLW bitfield remains unchanged. In other words it is not possible to configure NBLW and set DCAL at the same time.

Reading NBLW bits returns the last value written to NBLW.

0x00: All the 32 bits of the last data written are valid message bits, that is M[31:0]

0x01: Only one bit of the last data written (after swapping) is valid, that is M[0]

0x02: Only two bits of the last data written (after swapping) are valid, that is M[1:0]

0x03: Only three bits of the last data written (after swapping) are valid that is M[2:0]

...

0x1F: Only 31 bits of the last data written (after swapping) are valid that is M[30:0]

Note:

25.6.4 HASH digest registers

These registers contain the message digest result defined as follows:

- HASH_HR0, HASH_HR1, HASH_HR2, HASH_HR3 and HASH_HR4 registers return the SHA-1 digest result.
- HASH_HR0 to HASH_HR6 registers return the SHA-224 digest result.
- HASH_HR0 to HASH_HR7 registers return the SHA-256 digest result.

In all cases, the digest most significant bit is stored in HASH_HR0[31], and unused HASH_HRx registers reads as zero.

If a read access to one of these registers is performed while the hash core is calculating an intermediate digest or a final message digest (DCIS bit equals 0), then the read operation returns zeros.

Note: When starting a digest computation for a new message (by setting the INIT bit), HASH_HRx registers are forced to their reset values.

HASH_HR0 to HASH_HR4 registers can be accessed through two different addresses (register aliasing).

HASH aliased digest register x (HASH_HRAx)

Address offset: 0x00C + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

The content of the HASH_HRAx registers is identical to the one of the HASH_HRx registers located at address offset 0x310.

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

Bits 31:0 **Hx[31:0]:** Hash data x
Refer to *Section 25.6.4: HASH digest registers* introduction.

HASH digest register x (HASH_HRx)

Address offset: 0x310 + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

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

Bits 31:0 **Hx[31:0]:** Hash data x
Refer to *Section 25.6.4: HASH digest registers* introduction.

HASH supplementary digest register x (HASH_HRx)

Address offset: 0x310 + 0x4 * x, (x = 5 to 7)

Reset value: 0x0000 0000

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

Bits 31:0 **Hx[31:0]:** Hash data x
Refer to *Section 25.6.4: HASH digest registers* introduction.

25.6.5 HASH interrupt enable register (HASH_IMR)

Address offset: 0x020

Reset value: 0x0000 0000

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DCIE:** Digest calculation completion interrupt enable
0: Digest calculation completion interrupt disabled
1: Digest calculation completion interrupt enabled.

Bit 0 **DINIE:** Data input interrupt enable
0: Data input interrupt disabled
1: Data input interrupt enabled

25.6.6 HASH status register (HASH_SR)

Address offset: 0x024

Reset value: 0x0011 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBWE[4:0]				
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DINNE	Res.	NBWP[4:0]					Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
r		r	r	r	r	r						r	r	rc_w0	rc_w0

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **NBWE[4:0]:** Number of words expected
This bitfield reflects the number of words in the message that must be pushed into the FIFO to trigger a partial computation. NBWE is decremented by 1 when a write access is performed to the HASH_DIN register.
NBWE is set to the expected block size +1 in words (0x11) when INIT bit is set in HASH_CR. It is set to the expected block size (0x10) when the partial digest calculation ends.

Bit 15 **DINNE:** DIN not empty
This bit is set when the HASH_DIN register holds valid data (that is after being written at least once). It is cleared when either the INIT bit (initialization) or the DCAL bit (completion of the previous message processing) is set.
0: No data are present in the data input buffer
1: The input buffer contains at least one word of data

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **NBWP[4:0]:** Number of words already pushed

This bitfield is the exact number of words in the message that have already been pushed into the FIFO. NBWP is incremented by 1 when a write access is performed to the HASH_DIN register.

When a digest calculation starts, NBWP is updated to NBWP- block size (in words), and NBWP goes to zero when the INIT bit is set.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy bit

0: No block is currently being processed
1: The hash core is processing a block of data

Bit 2 **DMAS:** DMA Status

This bit provides information on the DMA interface activity. It is set with DMAE and cleared when DMAE = 0 and no DMA transfer is ongoing. No interrupt is associated with this bit.

0: DMA interface is disabled (DMAE = 0) and no transfer is ongoing
1: DMA interface is enabled (DMAE = 1) or a transfer is ongoing

Bit 1 **DCIS:** Digest calculation completion interrupt status

This bit is set by hardware when a digest becomes ready (the whole message has been processed). It is cleared by writing it to 0 or by setting the INIT bit in the HASH_CR register.

0: No digest available in the HASH_HRx registers (zeros are returned)

1: Digest calculation complete, a digest is available in the HASH_HRx registers. An interrupt is generated if the DCIE bit is set in the HASH_IMR register.

Bit 0 **DINIS:** Data input interrupt status

This bit is set by hardware when the FIFO is ready to get a new block (16 locations are free). It is cleared by writing it to 0 or by writing the HASH_DIN register.

When DINIS = 0, HASH_CSRx registers reads as zero.

0: Less than 16 locations are free in the input buffer
1: A new block can be entered into the input buffer. An interrupt is generated if the DINIE bit is set in the HASH_IMR register.

25.6.7 HASH context swap registers

These registers contain the complete internal register states of the hash processor. They are useful when a suspend/resume operation has to be performed because a high-priority task needs to use the hash processor while it is already used by another task.

When such an event occurs, the HASH_CSRx registers have to be read and the read values have to be saved in the system memory space. Then the hash processor can be used by the preemptive task. When the hash computation is complete, the saved context can be read from memory and written back into the HASH_CSRx registers.

HASH_CSRx registers can be read only when DINIS equals to 1, otherwise zeros are returned.

HASH context swap register x (HASH_CSRx)

Address offset: 0x0F8 + 0x4 * x, (x = 0 to 53)

Reset value: 0x0022 0002 (HASH_CSR0)

Reset value: 0x0000 0000 (others)

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

Bits 31:0 **CSx[31:0]:** Context swap x

Refer to *Section 25.6.7: HASH context swap registers* introduction.

25.6.8 HASH register map

Table 203. HASH1 register map and reset values

Offset	Register name	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
0x000	HASH_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		ALGO[1:0]		LKEY	Res.	Res.	MDMAT	DINNE	NBW[3:0]				Res.	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
	Reset value												0	0	0	0	0			0	0	0	0	0	0		0	0	0	0	0		
0x004	HASH_DIN	DATAIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	HASH_STR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
	Reset value																								0				0	0	0	0	0
0x00C + 0x4 * x, (x = 0 to 4) Last address: 0x01C	HASH_HRAx	Hx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	HASH_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCIE	DINIE
	Reset value																															0	0
0x024	HASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		NBWE[4:0]					DINNE		NBWP[4:0]					Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
	Reset value												1	0	0	0	1	0		0	0	0	0	0						0	0	0	1
Reserved																																	
0x0F8	HASH_CSR0	CSR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 203. HASH1 register map and reset values (continued)

Offset	Register name	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
0x0FC	HASH_CSR1	CSR1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100	HASH_CSR2	CSR2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x1CC	HASH_CSR53	CSR53[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reserved																																	
0x310 + 0x4 * x, (x = 0 to 7) Last address: 0x32C	HASH_HRx	Hx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

26 On-the-fly decryption engine (OTFDEC)

This section only applies to STM32WBA25 devices.

26.1 OTFDEC introduction

OTFDEC allows on-the-fly decryption of the AHB traffic based on the read request address information. Four independent and non-overlapping encrypted regions can be defined in OTFDEC.

OTFDEC uses AES-128 in counter mode to achieve the lowest possible latency. As a consequence, each time the content of one encrypted region is changed, the entire region must be re-encrypted with a different cryptographic context (key or initialization vector). This constraint makes OTFDEC suitable to decrypt read-only data or code, stored in external NOR flash.

Note: When OTFDEC is used in conjunction with OCTOSPI, it is mandatory to access the flash memory using the memory-mapped mode of the flash memory controller.

When security is enabled in the product, OTFDEC can be programmed only by a secure host.

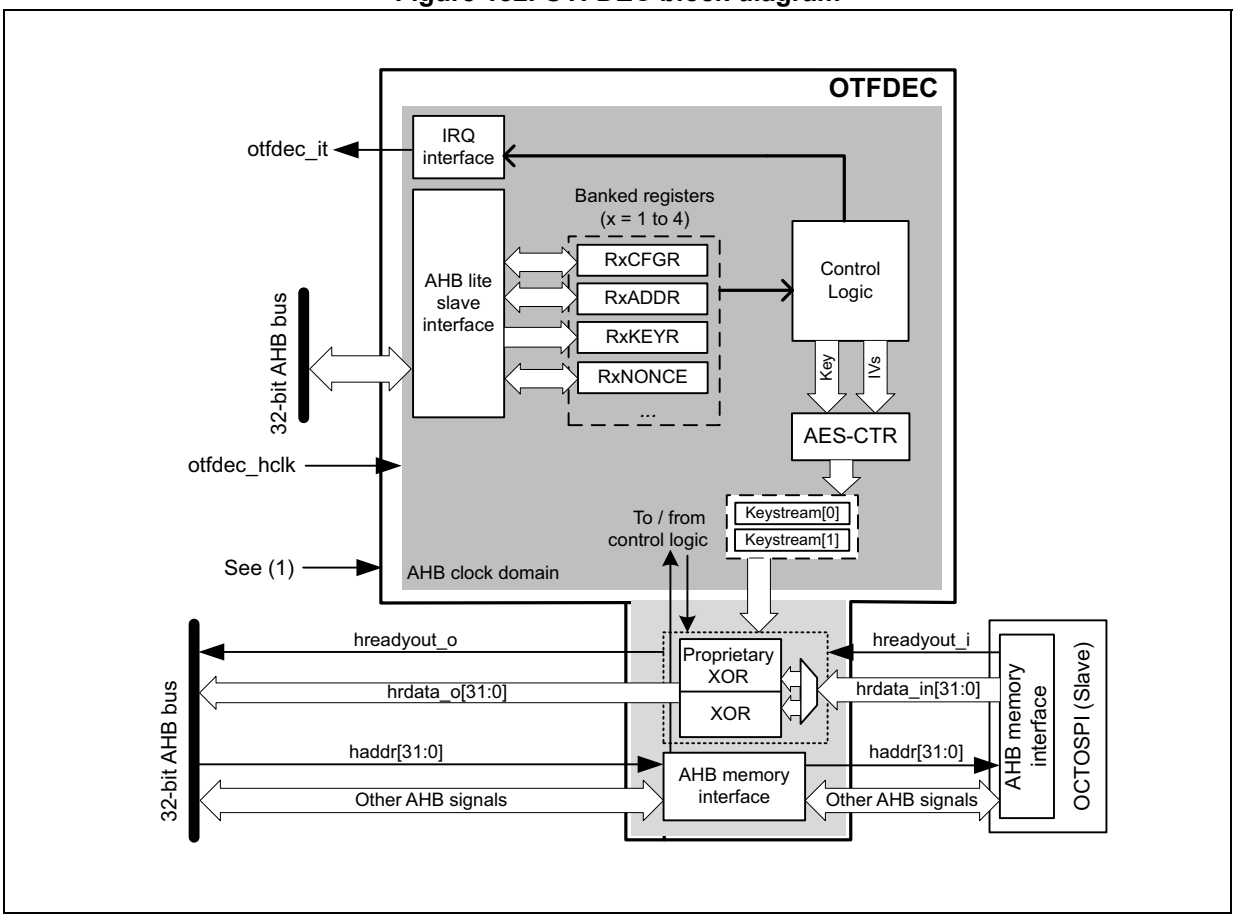
26.2 OTFDEC main features

- On-the-fly 128-bit decryption during the OCTOSPI Memory-mapped read operations (single or multiple).
  - Use of AES in counter (CTR) mode, with two 128-bit keystream buffers
  - Support for any read size
  - Physical address of the reads used for the encryption/decryption
- Up to four independent encrypted regions
  - Granularity of the region definition: 4096 bytes
  - Region configuration write-locking mechanism
  - Each region has its own 128-bit key, two bytes firmware version, and eight bytes application-defined nonce. At least one of those must be changed each time an encryption is performed by the application.
- Encryption keys confidentiality and integrity protection
  - Write-only registers, with software locking mechanism
  - Availability of 8-bit CRC as public key information
- Support for OCTOSPI pre-fetching mechanism
- Possibility to select an enhanced encryption mode to add a proprietary layer of protection on top of AES stream cipher (execute only)
- Privileged-aware AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
- Secure only programming if TrustZone security is enabled in the product
- Encryption mode

26.3 OTFDEC functional description

26.3.1 OTFDEC block diagram

Figure 132. OTFDEC block diagram



1. otfdec_tzen

26.3.2 OTFDEC internal signals

Table 204 describes a list of useful to know internal signals available at OTFDEC level, not at the product level (on pads).

Table 204. OTFDEC internal input/output signals

Signal name	Signal type	Description
otfdec_hclk	Digital input	AHB bus clock
otfdec_it	Digital output	OTFDEC global interrupt request
otfdec_tzen	Digital input	OTFDEC TrustZone enable, controlling TrustZone features of the peripheral (TZEN)

The TZEN option bit in FLASH is used to activate TrustZone in the device.

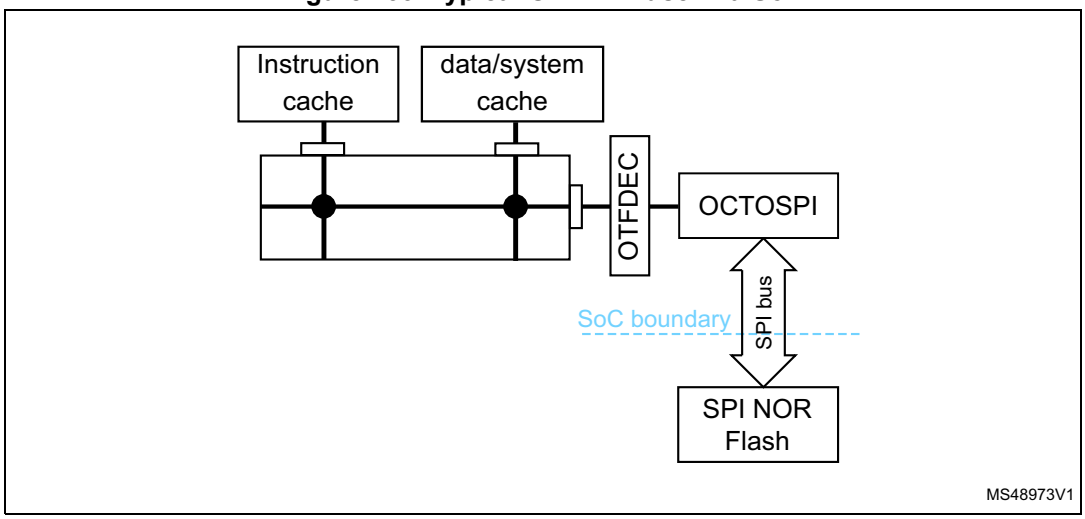
- TZEN = 1: TrustZone security is enabled in the product.
- TZEN = 0: TrustZone security is disabled in the product.

26.3.3 OTFDEC on-the-fly decryption

Introduction

Typical usage for OTFDEC is shown on *Figure 133*.

Figure 133. Typical OTFDEC use in a SoC



Original purpose of OTFDEC is to protect the confidentiality of read-only firmware libraries stored in external SPI NOR flash devices.

A special locking scheme is available in OTFDEC in order to protect the integrity of the decryption keys and also to protect the other configurations against software denial of services attacks. OTFDEC access to most registers can be made privileged-only by setting PRIV bit in OTFDEC_PRIVCFGR register. OTFDEC is only writeable by TrustZone CPU, when TrustZone security is activated.

When OTFDEC is used in conjunction with OCTOSPI, it is mandatory to read the flash memory using the Memory-mapped mode of the flash controller.

On top of decrypting on-the-fly, OTFDEC can also encrypt 32-bit word at a time (see *Section 26.5.3: Encrypting for OTFDEC* for more details).

OTFDEC architecture

OTFDEC analyzes all AHB read transfers on the associated AHB bus. If the read request is within one of the four regions programmed in OTFDEC, the control logic triggers a keystream computation based on AES algorithm in counter mode. This keystream is then used to decrypt on-the-fly the data present in the read transfer from the OCTOSPI AHB master, tying low the HREADYOUT signal of this master while the keystream information is being computed (this takes up to 11 cycles). Any accesses outside the enabled OTFDEC regions belong to a non-encrypted region.

Each OTFDEC region is programmed through OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR, OTFDEC_RxNONCER and

OTFDEC_RxKEYR registers, where x = 1 to 4. In OTFDEC_RxCFGR, the MODE bits define the OTFDEC operating mode (standard or enhanced encryption).

Granularity for the region determination is 4096 bytes.

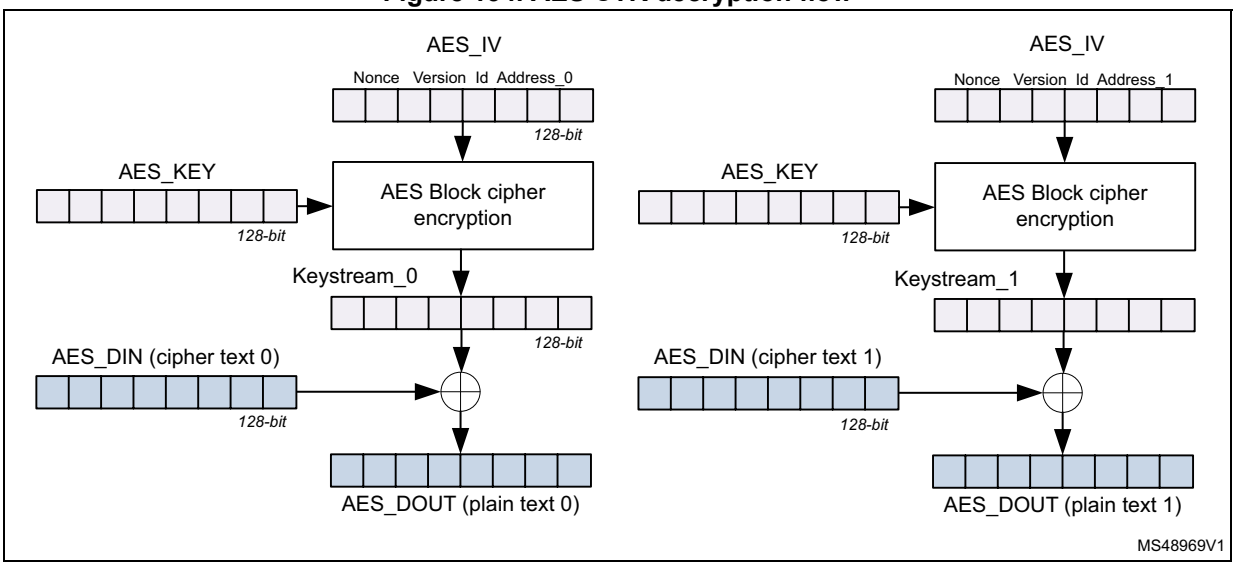
Note: Although OTFDEC does not prevent region overlapping, it is not a valid programming and it must be avoided by application software.

OTFDEC can decrypt incremental or wrap bursts only if they do not cross the 4096-byte aligned address boundaries.

26.3.4 OTFDEC usage of AES in counter mode decryption

Figure 134 shows how OTFDEC uses industry standard Advanced Encryption Standard (AES) algorithm in counter chaining mode. This mode is specified by NIST in *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*.

Figure 134. AES CTR decryption flow



Every 128-bit data block, a special keystream information is computed using AES block cipher, as defined below:

- initialization vector AES_IV[127:0] = RxNONCER1[31:0] || RxNONCER0[31:0] || 0b0000 0000 0000 0000 || RxCFGR[31:16] || 0b00 || (x-1) || ReadAddress[31:4]
- key material AES_KEY[127:0] = R*x*KEYR3[31:0] || R*x*KEYR2[31:0] || R*x*KEYR1[31:0] || R*x*KEYR0[31:0]

Note: Above x is the RegionID of the selected encrypted region (x=1 to 4).
ReadAddress is the AHB address of the encrypted data block, modulo 128-bit.

Resulting 128-bit keystream is XORed with 128-bit cipher text data to produce the 128-bit clear text data.

- AES_DIN and AES_DOUT data blocks are constructed following the rule below (“|” represents a binary concatenation):
  AES_Dx[127:0]= AHB_word(@ | 0xC)[31:0] | AHB_word(@ | 0x8)[31:0] | AHB_word(@ | 0x4)[31:0] || AHB_word(@ | 0x0)[31:0], where @ is the hexadecimal address used to compute the keystream (ReadAddress[31:4] above).

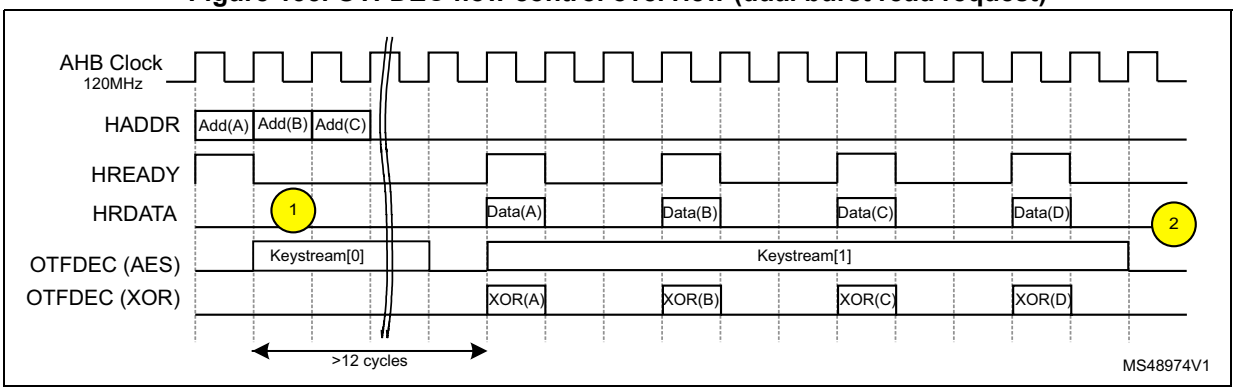
When the read request is not within an encrypted region, or the decryption is not enabled in this region, the AHB data is not changed.

Note: When the application sets the MODE bitfield to 11 in OTFDEC_RxCFGR, an additional layer of protection is added on top of the AES stream cipher. This enhanced encryption mode can only be used with instructions (execute-only region).

26.3.5 Flow control management

Figure 135 shows how OTFDEC manages one INCR4 AHB burst that corresponds to one 128-bit AES data block.

Figure 135. OTFDEC flow control overview (dual burst read request)



with the following notes:

1. OTFDEC enforces HREADY signal from the AHB master low as it is not ready to decrypt data (keystream computation).
2. Thanks to the keystream buffer, OTFDEC can be ready to process a new batch of data within 12 cycles in this configuration (120 MHz AHB clock, 104 MHz SPI bus delivering 2 bytes per SPI clock).

26.3.6 OTFDEC error management

OTFDEC automatically manages errors defined as below:

- Illegal read to OTFDEC_RxKEYR registers
- Illegal write to OTFDEC_RxKEYR registers while CONFIGLOCK or KEYLOCK = 1 in OTFDEC_RxCFGR, while the access is secure. If the security is disabled in the product, the same error occurs when the access is nonsecure.
- Illegal write to OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR or OTFDEC_RxNONCER registers while CONFIGLOCK = 1 in OTFDEC_RxCFGR (x = 1 to 4), while the access is secure. If the security is disabled in the product the same error occurs when the access is nonsecure.
- Illegal read to an execute-only region (MODE[1:0] = 11). Such illegal request returns 0x0, without bus error.
- Execution request to a region while encryption is enabled (ENC = 1). The request returns 0x0, without bus error.
- Key error: read request to an encrypted region while its key registers are null or not properly initialized (KEYCRC = 0x0). Source of the error can be an incorrect key loading sequence (see KEYCRC in OTFDEC_RxCFGR) or it can be an abort event

(tamper detection, unauthorized debug connection, untrusted boot, RDP level regression). Such read request returns 0x0, without bus error.

- Write to any registers while the access is nonsecure, if TrustZone security is enabled in the product.

  This last error is managed and cleared through TrustZone interrupt controller, as described in the GTZC section of the product reference manual.

For these errors (except the last one), an interrupt can be generated if the SEIE, XONEIE or KEIE bit is set in OTFDEC_IER register (see *Section 26.4*).

Note: After a key error, OTFDEC keys must be properly initialized again, and a reset of OTFDEC may be needed if registers are locked.

26.4 OTFDEC interrupts

There are three independent maskable interrupt sources generated by the OTFDEC, signaling following security events:

- Illegal read or write access to keys (SEIF flag), see *Section 26.3.6*
- Illegal write to a region configuration while CONFIGLOCK = 1 (SEIF flag), see *Section 26.3.6*
- Read access to an execute-only region (MODE[1:0] = 11), triggering the XONEIF flag
- Executing while encryption is enabled (XONEIF flag)
- Key error (encrypted regions read as zero) triggering the KEIF flag, see *Section 26.3.6*.

Interrupt sources are connected to the same global interrupt request signal.

OTFDEC interrupt sources can be enabled/disabled by setting the corresponding SEIE, XONEIE or KEIE bit in OTFDEC_IER, as described in *Table 205*. Status of the interrupt event is found in OTFDEC_ISR, and this event can be cleared using OTFDEC_ICR.

Table 205. OTFDEC interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit	Interrupt clear method
OTFDEC	Security error	SEIF	SEIE	Set SEIF in OTFDEC_ICR
	Execute-only Execute while encryption	XONEIF	XONEIE	Set XONEIF in OTFDEC_ICR
	Key error	KEIF	KEIE	Set KEIF in OTFDEC_ICR

1. The event flags are found in the OTFDEC_ISR register.

26.5 OTFDEC application information

26.5.1 OTFDEC initialization process

Introduction

One key aspect of OTFDEC is the trusted initialization of its registers, as it involves secret keys. Two trusted initialization schemes are recommended here below.

Note: Those sequences are for production code, as during firmware development, it is not always recommended to lock the key or the region configuration.

Writes to configuration registers are effective when the configuration locks allow it, even if the region is enabled.

Initialization scheme 1: one key for all regions

In this scheme, one entity owns the secret key used to decrypt the four protected regions. The recommended OTFDEC configuration sequence is described below:

1. For *x* = 1 to 4, write the correct MODE[1:0] value in OTFDEC_RxCFGR.
2. For *x* = 1 to 4, program OTFDEC_RxKEYR registers using the sequence described in KEYCRC (to have a valid CRC). Warning as key registers are write only.
3. For *x* = 1 to 4, check the key CRC. If OK, set KEYLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (key registers in this region *x* are no more writable).
4. To do to decrypt a region x (task that does not necessarily have to be performed by the entity that owns the decryption keys):
   a) Verify if the key CRC corresponds to the encrypted binary stored in the region.
   b) Fill the detailed information corresponding to this binary (nonce, start address, end address, version number).
   c) Enable decryption of this region using REG_EN.
   d) Set CONFIGLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (the region configuration is no more writable).

Caution: For a given region, when MODE bits are changed, the key registers and associated CRC are cleared by hardware. As a consequence, step 1 above must be done before step 2, and MODE bits must not be modified after step 2.

Initialization scheme 2: one key per region

In this scheme, one entity can own the secret used to decrypt one (or more) protected region. The recommended OTFDEC configuration sequence is described below:

1. To do to decrypt a region x (this task **must** be performed by the entity that owns the corresponding key):
   a) Write the correct MODE[1:0] value in OTFDEC_R*x*CFGR.
   b) Program OTFDEC_RxKEYR registers using the sequence described in KEYCRC (to have a valid CRC). Warning as key registers are write only.
   c) Check the key CRC. If OK, set KEYLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (key registers are no more writable).
   d) Fill the detailed information corresponding to the protected firmware (nonce, start address, end address, version number).
   e) Enable decryption of this region using REG_EN.
   f) Set CONFIGLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (the region configuration is no more writable).

Caution: For a given region, when MODE bits are changed, the key registers and associated CRC are cleared by hardware. As a consequence step a) above must be done before step b), and MODE bits must not be modified after step b).

26.5.2 OTFDEC and power management

Each time OTFDEC is reset, the correct key loading sequence described in *Section 26.5.1* must be performed (in this case KEYCRC = 0 in OTFDEC_RxCFGR).

It is recommended for application software to verify this point each time OTFDEC is reset by hardware.

26.5.3 Encrypting for OTFDEC

Code and data standard encryption

OTFDEC uses standard AES in counter mode when processing a binary stored in a protected region with MODE[1:0] = 10. When this mode is selected, any AES compatible hardware accelerator or library can be used to encrypt those protected libraries. OTFDEC can be used as well, as described in enhanced encryption section below (with MODE[1:0] = 10).

Definition and endianness of the AES inputs and outputs are defined in *Section 26.3.4: OTFDEC usage of AES in counter mode decryption*.

Enhanced encryption with OTFDEC

OTFDEC uses a proprietary layer of protection on top of the standard AES in counter mode when processing a code stored in a protected region with MODE[1:0] = 11.

Enhanced encryption mode can be used to increase the robustness against tampering.

Recommended sequence to encrypt using OTFDEC is described below:

1. The application in charge of the encryption sets the ENC bit in OTFDEC_CR. This application must run in TrustZone secure mode when TrustZone security is enabled in the product. If PRIV bit is set in OTFDEC_PRIVCFGR, this application must be privileged.
2. Encryption application initializes OTFDEC as described in *Section 26.5.1: OTFDEC initialization process*. OCTOSPI must also be properly clocked, so that OTFDEC is fully functional in encryption mode. This step can also be done before step 1.
3. Encryption application writes 32-bit of clear-text data at the expected protected address, then reads it back encrypted at the same address to store it in RAM. Note that this data stays inside the device, as it is intercepted by OTFDEC in encryption mode.
4. Encryption application goes back to previous step (changing the address) until the whole binary is processed.
5. Encryption application clears the ENC bit in OTFDEC_CR. Another application can then take the encrypted binary and flash it to the correct address in external flash.

There are few important notes about this procedure:

- Encryption granularity is 32-bit (single 32-bit access is mandatory).
- While ENC bit is set, reads to non-encrypted regions return normal data (such as no encryption nor decryption). While in encryption mode, no access to OCTOSPI (including registers) must be done. This is because the OTFDEC cuts the communication with OCTOPSI while ENC bit is set.
- OTFDEC does not support execution while ENC = 1 (only encrypted data reads). Upon illegal execution detection a XONEIF flag is raised and zero is returned.

26.5.4 OTFDEC key CRC source code

Below is the CRC source code that can be used to compare with the result of the computation provided by OTFDEC in KEYCRC bitfield after loading the keys in OTFDEC_RxKEYR registers.

```
uint8_t getCRC(uint32_t * keyin)
{
  const uint8_t CRC7_POLY = 0x7;
  const uint32_t key_strobe[4] = {0xAA55AA55, 0x3, 0x18, 0xC0};
  uint8_t  i, j, k, crc = 0x0;
  uint32_t  keyval;

  for (j = 0; j < 4; j++)
  {
    keyval = *(keyin+j);
    if (j == 0)
    {
      keyval ^= key_strobe[0];
    }
    else
    {
      keyval ^= (key_strobe[j] << 24) | (crc << 16) | (key_strobe[j] << 8)
| crc;
    }

    for (i = 0, crc = 0; i < 32; i++)
    {
      k = (((crc >> 7) ^ (keyval >> (31-i))&0xF)) & 1;
      crc <<= 1;
      if (k)
      {
        crc ^= CRC7_POLY;
       }
    }
    crc^=0x55;
  }
  return crc;
}
```

26.6 OTFDEC registers

26.6.1 OTFDEC control register (OTFDEC_CR)

Address offset: 0x0

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **ENC**: Encryption mode bit

When this bit is set, OTFDEC is used in encryption mode, during which application can write clear text data then read back encrypted data. When this bit is cleared (default), OTFDEC is used in decryption mode, during which application only read back decrypted data. For both modes, cryptographic context (keys, nonces, firmware versions) must be properly initialized. When this bit is set, only data accesses are allowed (zeros are returned otherwise, and XONEIF is set). When MODE = 11, enhanced encryption mode is automatically selected.

0: OTFDEC working in decryption mode

1: OTFDEC working in encryption mode

Note: When ENC bit is set, no access to OCTOSPI must be done (registers and Memory-mapped region).

26.6.2 OTFDEC privileged access control configuration register (OTFDEC_PRIVCFGR)

Address offset: 0x10

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

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

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PRIV:** Privileged access protection.

0: No additional protection is added on OTFDEC register accesses.

1: An additional protection is added when accessing all registers except OTFDEC_PRIVCFGR:

- Unprivileged read accesses to registers return zeros
- Unprivileged write accesses to registers are ignored.

Note: This bit can only be written in privileged mode. There is no limitations on reads.

26.6.3 OTFDEC region x configuration register (OTFDEC_RxCFGR)

Address offset: 0x20 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

Writes are ignored if CONFIGLOCK bit is set to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_VERSION[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYCRC[7:0]								Res.	Res.	MODE[1:0]		Res.	KEYLOCK	CONFIGLOCK	REG_EN
r	r	r	r	r	r	r	r			rw	rw		rs	rs	rw

Bits 31:16 **REG_VERSION[15:0]**: region firmware version

This 16-bit bitfield must be correctly initialized before the region corresponding REG_EN bit is set in OTFDEC_RxCFGR.

Bits 15:8 **KEYCRC[7:0]**: region key 8-bit CRC

When KEYLOCK = 0, KEYCRC bitfield is automatically computed by hardware while loading the key of this region in this exact sequence: KEYR0 then KEYR1 then KEYR2 then finally KEYR3 (all written once). A new computation starts as soon as a new valid sequence is initiated, and KEYCRC is read as zero until a valid sequence is completed.

When KEYLOCK = 1, KEYCRC remains unchanged until the next reset.

CRC computation is an 8-bit checksum using the standard CRC-8-CCITT algorithm $X^8 + X^2 + X + 1$ (according the convention). Source code is available in *Section 26.5.4*.

This field is read only.

Note: CRC information is updated only after the last bit of the key has been written.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MODE[1:0]**: operating mode

This bitfield selects the OTFDEC operating mode for this region:

10: All read accesses are decrypted (instruction or data).

11: Enhanced encryption mode is activated, and only instruction accesses are decrypted

Others: Reserved

When MODE ≠ 11, the standard AES encryption mode is activated.

When either of the MODE bits are changed, the region key and associated CRC are zeroed.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **KEYLOCK**: region key lock

0: Writes to this region KEYRx registers are allowed.

1: Writes to this region KEYRx registers are ignored until next OTFDEC reset. KEYCRC bitfield is locked.

Note: This bit is set once: if this bit is set, it can only be reset to 0 if the OTFDEC is reset.

Bit 1 **CONFIGLOCK**: region config lock

0: Writes to this region OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR and OTFDEC_RxNONCERy registers are allowed.

1: Writes to this region OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR and OTFDEC_RxNONCERy registers are ignored until next OTFDEC reset.

Note: This bit is set once. If this bit is set, it can only be reset to 0 if OTFDEC is reset. Setting this bit forces KEYLOCK bit to 1.

Bit 0 **REG_EN**: region on-the-fly decryption enable

0: On-the-fly decryption is disabled for this region.

1: On-the-fly decryption is enabled for this region. Data are XORed with the corresponding keystream.

Note: Garbage is decrypted if region context (version, key, nonce) is not valid when this bit is set.

26.6.4 OTFDEC region x start address register (OTFDEC_RxSTARTADDR)

Address offset: 0x24 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:0 **REG_START_ADDR[31:0]**: Region AHB start address

This register must be written before the region corresponding REG_EN bit in the OTFDEC_RxCFGR register is set.

Writing to this register is discarded if performed while the region CONFIGLOCK bit in the OTFDEC_RxCFGR register is set.

Note: When determining the region the first 12 bits (LSB) and the last 4 bits (MSB) are ignored.

When this register is accessed in read the 4 MSB bits and the 12 LSB bits return zeros .

26.6.5 OTFDEC region x end address register (OTFDEC_RxENDADDR)

Address offset: 0x28 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0FFF

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:0 **REG_END_ADDR[31:0]**: Region AHB end address

This register must be written before the region corresponding REG_EN bit in the OTFDEC_RxCFGR register is set, and OTFDEC_RxENDADDR must be strictly greater than OTFDEC_RxSTARTADDR to be valid.

Writing to this register is discarded if performed while the region CONFIGLOCK bit in OTFDEC_RxCFGR is set.

Note: When determining the region the first 12 bits (LSB) and the last 4 bits (MSB) are ignored.

When this register is accessed in read the 4 MSB bits return zeros and the 12 LSB bits return ones.

26.6.6 OTFDEC region x nonce register 0 (OTFDEC_RxNONCER0)

Address offset: 0x2C + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:0 **REG_NONCE[31:0]**: Region nonce, bits [31:0]

This register must be written before the region corresponding REG_EN bit in OTFDEC_RxCFGR is set.

Writing is discarded in this register if performed while the region CONFIGLOCK bit in the OTFDEC_RxCFGR is set.

26.6.7 OTFDEC region x nonce register 1 (OTFDEC_RxNONCER1)

Address offset: 0x30 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_NONCE[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_NONCE[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REG_NONCE[63:32]**: Region nonce, bits [63:32]

Refer to the OTFDEC_RxNONCER0 register for description of the NONCE[63:0] bitfield.

26.6.8 OTFDEC region x key register 0 (OTFDEC_RxKEYR0)

Address offset: 0x34 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:0 **REG_KEY[31:0]**: Region key, bits [31:0]

This register must be written before the region corresponding REG_EN bit in OTFDEC_RxCFGR is set.

Reading this register returns a zero value. Writing to this register is discarded if performed while the region CONFIGLOCK or KEYLOCK bit is set in the OTFDEC_RxCFGR.

Note: When application successfully changes MODE bits in OTFDEC*_RxCFGR and* OTFDEC*_RxKEYR, and associated KEYCRC are erased.*

26.6.9 OTFDEC region x key register 1 (OTFDEC_RxKEYR1)

Address offset: 0x38 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REG_KEY[63:32]**: Region key, bits [63:32]
Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

26.6.10 OTFDEC region x key register 2 (OTFDEC_RxKEYR2)

Address offset: 0x3C + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REG_KEY[95:64]**: Region key, bits [95:64]
Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

26.6.11 OTFDEC region x key register 3 (OTFDEC_RxKEYR3)

Address offset: 0x40 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REG_KEY[127:96]**: Region key, bits [127:96]

Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

26.6.12 OTFDEC interrupt status register (OTFDEC_ISR)

Address offset: 0x300

Reset value: 0x0000 0000

Unprivileged reads return zero if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	XONEIF	SEIF
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIF:** Key error interrupt flag status

This bit is set by hardware and read only by application. The bit is set when a read access occurs on an encrypted region, while its key registers is null or not properly initialized (KEYCRC = 0x0).

This bit is cleared when the application sets in OTFDEC_ICR the corresponding bit to 1.

0: OTFDEC operates properly.

1: Read access detected on an enabled encrypted region with its key registers null or not properly initialized (KEYCRC = 0x0). OTFDEC returns a zeroed value for the read, and an optional interrupt is generated if bit KEIE is set to 1 in OTFDEC_IER.

After KEIF is set any subsequent read to the region with bad key registers returns a zeroed value. This state remains until those key registers are properly initialized (KEYCRC not zero).

Bit 1 **XONEIF:** Execute-only execute-never error interrupt flag status

This bit is set by hardware and read only by application. This bit is set when a read access and not an instruction fetch is detected on any encrypted region with MODE bits set to 11. Lastly, XONEIF is also set when an execute access is detected while encryption mode is enabled.

This bit is cleared when application sets in OTFDEC_ICR the corresponding bit to 1.

0: No execute-only error status. No interrupt pending.

1: Read access detected on one region with MODE bits set to 11 or execute access detected while ENC = 1. OTFDEC returns a zeroed value for the illegal access, and an optional interrupt is generated if bit XONEIE is set to 1 in OTFDEC_IER.

Bit 0 **SEIF:** Security error interrupt flag status

This bit is set by hardware and read only by application. This bit is set when at least one security error has been detected.

This bit is cleared when application sets in OTFDEC_ICR the corresponding bit to 1.

0: No security error status. No interrupt pending.

1: Security error flag status, with interrupt pending. Actual interrupt generation is dependent on OTFDEC_IER corresponding bit SEIE.

26.6.13 OTFDEC interrupt clear register (OTFDEC_ICR)

Address offset: 0x304

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	XONEIF	SEIF
													w	w	w

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIF:** Key error interrupt flag clear
This bit is written by application, and always read as 0.
0: KEIF flag status is not affected.
1: KEIF flag status is cleared in OTFDEC_ISR.
Note: Clearing KEIF does not solve the source of the problem (bad key registers). To be able to access again any encrypted region, OTFDEC key registers must be properly initialized again.

Bit 1 **XONEIF:** Execute-only execute-never error interrupt flag clear
This bit is written by application, and always read as 0.
0: XONEIF flag status is not affected.
1: XONEIF flag status is cleared in OTFDEC_ISR.

Bit 0 **SEIF:** Security error interrupt flag clear
This bit is written by application, and always read as 0.
0: SEIF flag status is not affected.
1: SEIF flag status is cleared in OTFDEC_ISR.

26.6.14 OTFDEC interrupt enable register (OTFDEC_IER)

Address offset: 0x308

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIE	XONEIE	SEIE
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIE:** Key error interrupt enable

This bit is read and written by application. It controls the OTFDEC interrupt generation when KEIF flag status is set.

0: Interrupt generation on key error flag KEIF is disabled (masked).

1: Interrupt generation on key error flag KEIF is enabled (not masked).

Bit 1 **XONEIE:** Execute-only execute-never error interrupt enable

This bit is read and written by application. It controls the OTFDEC interrupt generation when XONEIF flag status is set.

0: Interrupt generation on execute-only error XONEIF is disabled (masked).

1: Interrupt generation on execute-only error XONEIF is enabled (not masked).

Bit 0 **SEIE:** Security error interrupt enable

This bit is read and written by application. It controls the OTFDEC interrupt generation when SEIF flag status is set.

0: Interrupt generation on security error SEIF is disabled (masked).

1: Interrupt generation on security error SEIF is enabled (not masked).

26.6.15 OTFDEC register map

Table 206. OTFDEC register map and reset values

Offset	Register name	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
0x00	OTFDEC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC
	Reset value																																0
0x04-0x0C	Reserved	Reserved																															
0x10	OTFDEC_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV
	Reset value																																0
0x14-0x1C	Reserved	Reserved																															
0x20	OTFDEC_R1CFGR1	REG1_VERSION[15:0]																KEYCRC[7:0]									Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x24	OTFDEC_R1STARTADDR	REG1_START_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	OTFDEC_R1ENDADDR	REG1_END_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	OTFDEC_R1NONCER0	REG1_NONCE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	OTFDEC_R1NONCER1	REG1_NONCE[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	OTFDEC_R1KEYR0	REG1_KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 206. OTFDEC register map and reset values (continued)

Offset	Register name	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
0x38	OTFDEC_ R1KEYR1	REG1_KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	OTFDEC_ R1KEYR2	REG1_KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	OTFDEC_ R1KEYR3	REG1_KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44 - 0x4C	Reserved	Reserved																															
0x50	OTFDEC_ R2CFGR	REG2_VERSION[15:0]																KEYCRC[7:0]									Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x54	OTFDEC_ R2STARTADDR	REG2_START_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x58	OTFDEC_ R2ENDADDR	REG2_END_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	OTFDEC_ R2NONCER0	REG2_NONCE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	OTFDEC_ R2NONCER1	REG2_NONCE[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x64	OTFDEC_ R2KEYR0	REG2_KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68	OTFDEC_ R2KEYR1	REG2_KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6C	OTFDEC_ R2KEYR2	REG2_KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70	OTFDEC_ R2KEYR3	REG2_KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74 - 0x7C	Reserved	Reserved																															
0x80	OTFDEC_ R3CFGR	REG3_VERSION[15:0]																KEYCRC[7:0]									Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x84	OTFDEC_ R3STARTADDR	REG3_START_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x88	OTFDEC_ R3ENDADDR	REG3_END_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 206. OTFDEC register map and reset values (continued)

Offset	Register name	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
0x8C	OTFDEC_ R3NONCER0	REG3_NONCE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x90	OTFDEC_ R3NONCER1	REG3_NONCE[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x94	OTFDEC_ R3KEYR0	REG3_KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x98	OTFDEC_ R3KEYR1	REG3_KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9C	OTFDEC_ R3KEYR2	REG3_KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA0	OTFDEC_ R3KEYR3	REG3_KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4 - 0xAC	Reserved	Reserved																															
0xB0	OTFDEC_ R4CFGR	REG4_VERSION[15:0]																KEYCRC[7:0]								Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0xB4	OTFDEC_ R4STARTADDR	REG4_START_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB8	OTFDEC_ R4ENDADDR	REG4_END_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBC	OTFDEC_ R4NONCER0	REG4_NONCE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC0	OTFDEC_ R4NONCER1	REG4_NONCE[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC4	OTFDEC_ R4KEYR0	REG4_KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC8	OTFDEC_ R4KEYR1	REG4_KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xCC	OTFDEC_ R4KEYR2	REG4_KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD0	OTFDEC_ R4KEYR3	REG4_KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D4- 0x2FC	Reserved	Reserved																															

Table 206. OTFDEC register map and reset values (continued)

Offset	Register name	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
0x300	OTFDEC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	XONEIF	SEIF
	Reset value																														0	0	0
0x304	OTFDEC_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	XONEIF	SEIF
	Reset value																														0	0	0
0x308	OTFDEC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIE	XONEIE	SEIE
	Reset value																														0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

27 Public key accelerator (PKA)

27.1 PKA introduction

PKA (public key accelerator) is intended for the computation of cryptographic public key primitives, specifically those related to RSA, Diffie-Hellmann or ECC (elliptic curve cryptography) over GF(p) (Galois fields). To achieve high performance at a reasonable cost, these operations are executed in the Montgomery domain.

For a given operation, the accelerator performs all computations with no further hardware or software elaboration required to process the inputs or the outputs.

For manipulating secrets, the PKA incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), required for SESIP and PSA security assurance level 3 target.

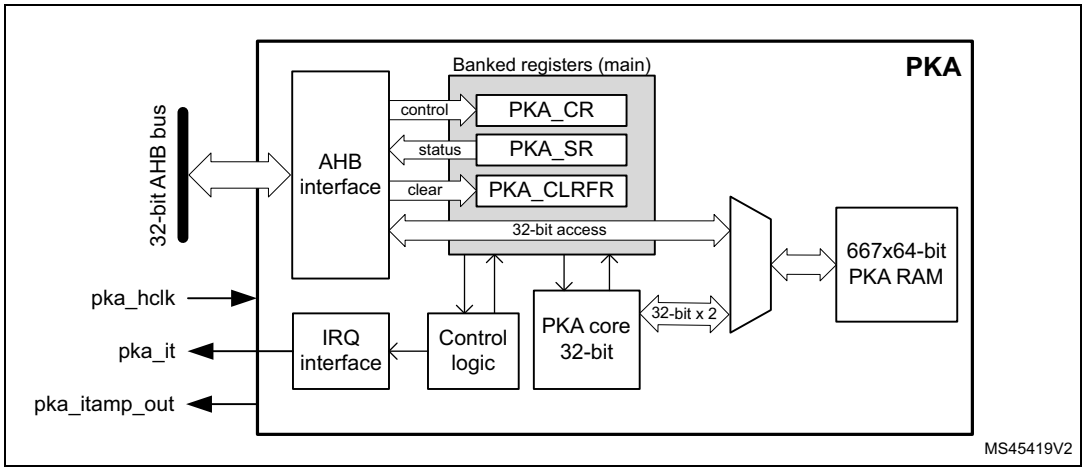
27.2 PKA main features

- Acceleration of RSA, DH and ECC over GF(p) operations, based on the Montgomery method for fast modular multiplications. More specifically:
  - RSA modular exponentiation, RSA chinese remainder theorem (CRT) exponentiation
  - ECC scalar multiplication, point on curve check, complete addition, double base ladder, projective to affine
  - ECDSA signature generation and verification
- Capability to handle operands up to 4160 bits for RSA/DH and 640 bits for ECC
- When manipulating secrets: protection against side-channel attacks (SCA), including differential power analysis (DPA), required for SESIP and PSA security assurance level 3 target.
  - Applicable to modular exponentiation, ECC scalar multiplication and ECDSA signature generation
- Arithmetic and modular operations such as addition, subtraction, multiplication, modular reduction, modular inversion, comparison, and Montgomery multiplication
- Built-in Montgomery domain inward and outward transformations
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)

27.3 PKA functional description

27.3.1 PKA block diagram

Figure 136. PKA block diagram



27.3.2 PKA internal signals

Table 207 lists the internal signals available at the PKA level, not necessarily available on product bonding pads.

Table 207. Internal input/output signals

Signal name	Signal type	Description
pka_hclk	Digital input	AHB bus clock
pka_it	Digital output	Public key accelerator IP global interrupt request
pka_itamp_out	Digital output	PKA internal tamper event signal to TAMP (XOR-ed), triggered when an unexpected fault occurs while PKA manipulates secrets, or when the programmed input point is not found on the input curve (ECDSA signature and ECC scalar multiplication only). This signal is asserted as soon as a fault is detected. When asserted, read access to PKA registers are reset to 0 and writes are ignored. The signal is de-asserted when PKA memory is cleared.

27.3.3 PKA reset and clocks

PKA is clocked on the AHB bus clock. When the PKA peripheral reset signal is released PKA_RAM is cleared automatically, taking 667 clock cycles. During this time the setting of bit EN in PKA_CR register is ignored. Once EN bit is set, refer to *Section 27.3.6* for details about PKA initialization sequence.

According to the security policy applied to the device, PKA RAM can also be reset following a tamper event. Refer to *Tamper detection and response* in the System security section (if applicable to this product).

27.3.4 PKA public key acceleration

Overview

Public key accelerator (PKA) is used to accelerate Rivest, Shamir and Adleman (RSA), Diffie-Hellman (DH) as well as ECC over prime field operations. Supported operand sizes is up to 4160 bits for RSA and DH, and up to 640 bits for ECC.

The PKA supports all non-singular elliptic curves defined over prime fields, that can be described with a short Weierstrass equation $y^2 = x^3 + ax + b$ (mod p). More information can be found in *Section 27.5.1*.

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.

A memory of 5336 bytes (667 words of 64 bits) called PKA RAM is used to provide initial data to the PKA, and to hold the results after computation is completed. Access is done though the PKA AHB interface.

PKA operating modes

The list of operations the PKA can perform is detailed in *Table 208* and *Table 209*, respectively, for integer arithmetic functions and prime field (Fp) elliptic curve functions.

Table 208. PKA integer arithmetic functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x01	000001	Montgomery parameter computation R^2 mod *n*	*Section 27.4.2*
0x0E	001110	Modular addition (A+B) mod n	*Section 27.4.3*
0x0F	001111	Modular subtraction (A-B) mod n	*Section 27.4.4*
0x10	010000	Montgomery multiplication (AxB) mod n	*Section 27.4.5*
0x00	000000	Modular exponentiation A^e mod n	*Section 27.4.6*
0x02	000010	Modular exponentiation A^e mod n (fast mode)	
0x03	000011	Modular exponentiation A^e mod n (protected)	*Section 27.4.6*
0x08	001000	Modular inversion A^{-1} mod n	*Section 27.4.7*
0x0D	001101	Modular reduction A mod n	*Section 27.4.8*
0x09	001001	Arithmetic addition A+B	*Section 27.4.9*
0x0A	001010	Arithmetic subtraction A-B	*Section 27.4.10*
0x0B	001011	Arithmetic multiplication AxB	*Section 27.4.11*
0x0C	001100	Arithmetic comparison (A=B, A>B, A<B)	*Section 27.4.12*
0x07	000111	RSA CRT exponentiation	*Section 27.4.13*

Table 209. PKA prime field (Fp) elliptic curve functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x28	101000	Point on elliptic curve Fp check	*Section 27.4.14*
0x20	100000	ECC scalar multiplication kP (protected)	*Section 27.4.15*
0x23	100011	ECC complete addition	*Section 27.4.18*
0x24	100100	ECDSA sign (protected)	*Section 27.4.16*
0x26	100110	ECDSA verification	*Section 27.4.17*
0x27	100111	ECC double base ladder	*Section 27.4.19*
0x2F	101111	ECC projective to affine	*Section 27.4.20*

Each of these operating modes has an associated code that has to be written to the MODE[5:0] bitfield in the PKA_CR register. If the application selects any value that is not documented below the write to MODE[5:0] bitfield is ignored, and an operation error (OPERRF) is triggered. When this happens, a new operation must be selected after the error is cleared.

Some operations in *Table 208* and *Table 209* are indicated as protected. Those operations are used when manipulating secret keys (modular exponentiation for RSA decryption, scalar multiplication and signature for ECC). Those secrets (protected against side channel attacks) and any intermediate values are automatically erased when PKA RAM is cleared at the end of the protected operations (BUSY goes low). They are also protected against side channel attacks.

Caution: For security reason it is very important to select protected modular exponentiation (MODE[5:0] = 0x3) when performing RSA decryption.

Montgomery space and fast mode operations

For efficiency reasons, the PKA internally performs modular multiply operations in the Montgomery domain, automatically performing inward and outward transformations.

As Montgomery parameter computation is time consuming, the application can decide to use a faster mode of operation during which the precomputed Montgomery parameter is supplied before starting the operation. The performance improvement is detailed in *Section 27.5.2: Computation times*.

The only operation using fast mode is the modular exponentiation (MODE[5:0] = 0x02).

27.3.5 Typical applications for PKA

Introduction

The PKA can be used to accelerate a number of public key cryptographic functions such as:

- RSA encryption and decryption
- RSA key finalization
- CRT-RSA decryption
- DSA and ECDSA signature generation and verification
- DH and ECDH key agreement

The following publications specify the listed functions:

- FIPS PUB 186-4, Digital Signature Standard (DSS), July 2013 by NIST
- PKCS #1, RSA Cryptography Standard, v1.5, v2.1 and v2.2. by RSA Laboratories
- IEEE1363-2000, IEEE Standard Specifications for Public-Key Cryptography, January 2000
- ANSI X9.62-2005, Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA), November 2005

Refer to these documents for full details. This document limits itself to describing principles.

RSA key pair

For the following RSA operations, a public key and a private key information are defined as follows:

- Alice transmits her public key (*n*, *e*) to Bob. Numbers *n* and *e* are very large positive integers.
- Alice keeps secret her private key d, also a very large positive integer. Alternatively, this private key can also be represented by a quintuple (p, q, dp, dq, qInv).

For more information, refer to the RSA specification.

RSA encryption/decryption principle

As recommended by the PKCS#1 specification, Bob, to encrypt message M using Alice's public key (*n*, *e*), must go through the following steps:

1. Compute the encoded message EM = ENCODE(M), where ENCODE is an encoding method.
2. Turn EM into an integer *m*, with $0 \leq m < n$ and (*m*, *n*) being coprimes.
3. Compute ciphertext $c = m^e \bmod n$.
4. Convert the integer c into a string ciphertext C.

Alice, to decrypt ciphertext c using her private key *d*, follows the steps indicated below:

1. Convert the ciphertext C to an integer ciphertext representative c.
2. If necessary, retrieve the prime factors (p, q) using (n, e, d) information, then compute phi = (p - 1) × (q - 1). Refer to NIST SP800-56B Appendix C for details.
3. Recover plaintext $m = c^d \bmod n = (m^e)^d \bmod n$. If the private key is the quintuple (p, q, dp, dq, qInv), then plaintext *m* is obtained by performing the operations:
   a) $m_1 = c^{dp} \bmod p$
   b) $m_2 = c^{dq} \bmod q$
   c) $h = qInv\ (m_1 - m_2) \bmod p$
   d) $m = m_2 + h\ q$
4. Convert the integer message representative *m* to an encoded message EM.
5. Recover message M = DECODE(EM), where DECODE is a decoding method.

To accelerate these operations with the PKA, use *Modular exponentiation* $A^e \bmod n$ if the private key is *d*, or *RSA CRT exponentiation* if the private key is the quintuple (p, q, dp, dq, qInv).

Note: The decoding operation and the conversion operations between message and integers are specified in PKCS#1 standard.

For the decryption process, the protected version of modular exponentiation (MODE[5:0] = 0x3) is strongly recommended for security reason. For the encryption process, the MODE[5:0] = 0x3 setting cannot be used as it requires the knowledge of the private key.

Elliptic curve selection

For the following ECC operations, the curve parameters are defined as follows:

- Curve corresponds to the elliptic curve field agreed among actors (Alice and Bob). Supported curves parameters are summarized in *Section 27.5.1: Supported elliptic curves*.
- G is the chosen elliptic curve base point (also known as generator), with a large prime order n (that is, $n \times G$ = identity element O).

ECDSA message signature generation

This section describes the ECDSA (elliptic curve digital signature algorithm) signature generation function principle.

Alice, to sign a message *m* using her private key integer d_A, goes through the following steps:

1. Calculate e = HASH(*m*), where HASH is a cryptographic hash function.
2. Let z be the L_n leftmost bits of e, where L_n is the bit length of the group order *n*.
3. Select a cryptographically secure random integer k where $0 < k < n$.
4. Calculate the curve point $(x_1, y_1) = k \times G$.
5. Calculate $r = x_1 \bmod n$. If $r = 0$ go back to step 3.
6. Calculate $s = k^{-1} (z + rd_A) \bmod n$. If $s = 0$ go back to step 3.
7. The signature is the pair (*r*, *s*).

The PKA accelerates the steps *4.* to *7.* by using:

- *ECDSA sign*, or
- All of the following operations:
  - *ECC Fp scalar multiplication* $k \times P$
  - *Modular reduction* A mod n
  - *Modular inversion* A^{-1} mod n
  - *Modular addition* and *Modular and Montgomery multiplication*

ECDSA signature verification

This section describes the ECDSA (elliptic curve digital signature algorithm) signature verification function principle.

Bob, to authenticate Alice's signature, must have a copy of her public key curve point Q_A. Bob can verify that Q_A is a valid curve point by going through the following steps:

1. Check that Q_A is not equal to the identity element O
2. Check that Q_A is on the agreed curve
3. Check that $n \times Q_A = O$.

Then Bob executes the following procedure:

1. Verify that r and s are integer in [1, n-1]
2. Calculate e = HASH(m), where HASH is the agreed cryptographic hash function
3. Let z be the L_n leftmost bits of e
4. Calculate $w = s^{-1}$ mod n
5. Calculate $u_1 = zw$ mod n and $u_2 = rw$ mod n
6. Calculate the curve point $(x_1, y_1) = u_1 \times G + u_2 \times Q_A$
7. The signature is valid if $r = x_1$ (mod n), it is invalid otherwise.

The PKA accelerates the steps *4.* to *7.* by using *ECDSA verification*.

27.3.6 PKA procedure to perform an operation

Enabling/disabling PKA

Setting the EN bit in the PKA_CR register enables the PKA peripheral. The PKA becomes available when the INITOK bit is set in the PKA_SR register. When EN = 0, the PKA peripheral is kept under reset, with its memory still accessible by the application through the AHB interface.

Note: When the PKA is in the process of clearing its memory, the EN bit cannot be set.

When setting the EN bit in PKA_CR, make sure that the MODE[5:0] bitfield value corresponds to an authorized PKA operation (see OPERRF in Section 27.3.7).

Clearing the EN bit while a calculation is in progress aborts the operation. In this case, the content of the PKA memory is not guaranteed, with the exception of the PKA modes 0x03, 0x20 and 0x24. For these operations, the PKA memory is cleared after abort, making the memory unavailable for 667 cycles. During this clearing time only the PKA registers can be accessed, with writes to EN bits ignored.

If the INITOK bit stays at 0, make sure that the RNG peripheral is clocked and properly initialized, then try to enable the PKA again.

Data formats

The format of the input data and the results in the PKA RAM are specified, for each operation, in *Section 27.4*.

Executing a PKA operation

To execute any of the supported PKA operations, proceed as follows:

1. Load initial data into the PKA internal RAM, which is located at address offset 0x400.
2. Write in the MODE[5:0] bitfield of PKA_CR register, specifying the operation which is to be executed and then assert the START bit, also in PKA_CR register.
3. Wait until the PROCENDF bit in the PKA_SR register is set, indicating that the computation is complete.
4. Read the result data from the PKA internal RAM, then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.

Note: When PKA is busy (BUSY = 1), any access by the application to PKA RAM is ignored, and the RAMERRF flag is set in PKA_SR.

Selecting an illegal or unknown operation in step *2* triggers an OPERRF error and prevents the PROCENDF flag from being set and the procedure step *3* from being completed. See *Section 27.3.7* for details.

Using precomputed Montgomery parameters (PKA fast mode)

As explained in *Section 27.3.4*, when computing many operations with the same modulus, it can be beneficial for the application to compute the corresponding Montgomery parameter only once (see, for example, *Section 27.4.5*). This is referred to as *fast mode*.

To use the PKA peripheral in fast mode, follow this recommended procedure:

1. Load the modulus size and value information into the PKA RAM. Refer to *Section 27.5.1* for detail.
2. Configure the PKA_CR register to *Montgomery parameter computation* mode (MODE[5:0] = 0x1), then assert the START bit.
3. Wait until the PROCENDF bit in the PKA_SR register is set. Then, read the corresponding Montgomery parameter from the PKA memory and clear the PROCENDF bit by setting PROCENDFC in PKA_CLRFR.
4. Continue with the required PKA operation, loading the Montgomery information R2 mod *m* on top of the regular input data. Relevant addresses are provided in *Section 27.4*.

27.3.7 PKA error management

The following error flags in the PKA_SR register are set to indicate errors that may occur when using the PKA:

- **ADDRERRF** flags that the access to PKA RAM falls outside the expected range.
- **RAMERRF** flags an AHB access to the PKA RAM occurring while the PKA core uses the PKA RAM. While the flag is set, reads form the PKA RAM return 0 and writes are ignored.
- **OPERRF** flags that the selected operating mode using MODE[5:0] bitfield is not listed in PKA operating modes (see the bitfield description), or PKA is running in limited mode

(see LMF bit in PKA_SR). While the flag is set, writes to the MODE[5:0] bitfield are ignored.

For each error flag, the PKA generates an interrupt if enabled with the corresponding bit in the PKA_CR register (see *Section 27.7* for details).

The ADDRERRF, OPERRF, and RAMERRF error flags are cleared by setting the ADDRERRFC, OPERRFC, and RAMERRFC bit in PKA_CLRFR, respectively. The OPERRF error flag must be cleared before a new operation is initiated via the PKA_CR register.

Note: The PKA can be re-initialized at any moment by clearing the EN bit in the PKA_CR register.

27.4 PKA operating modes

27.4.1 Introduction

The operations supported by the PKA are detailed in the following subsections, which define the format of both input data and results stored in the PKA RAM.

Warning: Before starting any operation, ensure that all input parameters to the PKA are valid and consistent, as the PKA assumes their correctness. Additionally, input parameters must not exceed the operand sizes specified in the operation tables.

The following information applies to all PKA operations:

- The PKA core processes 64-bit words in its RAM. Accordingly, in the PKA section of this document, 'word' denotes a 64-bit unit.
- When writing an element as input in the PKA RAM, append an additional 64-bit word of zeros after the most significant word, except when the operand size is fixed at one word.
- All RAM storage addresses refer to the least significant word of the data. To obtain the actual address, the application must add the base address of the PKA to the indicated offset.
- Supported operand sizes are:
  - **ROS (RSA operand size)**: Data size is calculated as *(rsa_size / 64) + 1* words, where *rsa_size* is the modulus length in bits. For example, a 1024-bit RSA operand corresponds to 17 words (1088 bits).
  - **EOS (ECC operand size)**: Data size is *(ecc_size / 64) + 1* words, where *ecc_size* is the prime modulus length in bits. For example, a 192-bit ECC operand corresponds to four words (256 bits).

  Note: Both ROS and EOS include the additional zero word.
- Unless indicated otherwise, all operands in the tables are integers.

Note: Any fractional results in the formulas must be rounded to the nearest integer, as the PKA core processes data in 64-bit words.

Note: The maximum ROS is 66 words (4160-bit maximum exponent size), while the maximum EOS is 11 words (640-bit max operand size).

Example 1: Understanding endianness in PKA memory during ECC Fp scalar multiplication preparation

Write the *x*-coordinate of point P for an ECC P256 curve (EOS = 5 words) by placing the least significant bit at bit 0 of the word at address offset 0x578, and the most significant bit at bit 63 of the word at address offset 0x590. Then, write a zero word (0x0000000000000000) at address offset 0x598.

Example 2: Writing the curve coefficient '*a*' for ECC Fp scalar multiplication

To write the value *a* = -3 for a curve with a modulus length of 224 bits (rounded up to five 64-bit words), write the PKA memory with data as in the following table:

Table 210. Example of *'a'* curve coefficient for ECC Fp scalar

RAM offset	Value	Note
0x410	0x0000000000000001	Curve coefficient *a* sign without extra word
0x418	0x0000000000000011	Value of |*a*| LSB
0x420	0x0000000000000000	-
0x428	0x0000000000000000	-
0x430	0x0000000000000000	Value of |*a*| MSB
0x438	0x0000000000000000	Additional zero word

27.4.2 Montgomery parameter computation

This function is used to compute the Montgomery parameter (R^2 mod n) used by the PKA to convert operands into the Montgomery residue system representation. This operation can be very useful for fast mode operation because passing the Montgomery parameter as input saves the time for its computation.

Note: This operation also supports ECC curves when specifying the prime modulus length and EOS size.

The following table summarizes operation instructions for Montgomery parameter computation.

Table 211. Montgomery parameter computation

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x01	PKA_CR	6 bits
	Modulus length	(in bits, 0 ≤ value < 4160)	RAM@0x408	64 bits
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	ROS
OUT	Result: R^2 mod *n*	-	RAM@0x620	

27.4.3 Modular addition

The following table summarizes operation instructions for modular addition operation that calculates *A* + *B* mod *n*.

Table 212. Modular addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0E	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < n$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < n$)	RAM@0xC68	
	Modulus value *n*	($n < 2^{4160}$)	RAM@0x1088	
OUT	Result: $A + B \bmod n$	($0 \le$ result $< n$)	RAM@0xE78	

27.4.4 Modular subtraction

Modular subtraction operation calculates:

- ***A* - *B* mod *n***, when $A \ge B$
- ***A* + *n* - *B* mod *n***, when $A < B$

The following table summarizes the operation instructions.

Table 213. Modular subtraction

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0F	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < n$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < n$)	RAM@0xC68	
	Modulus value *n*	($n < 2^{4160}$)	RAM@0x1088	
OUT	Result: $A - B \bmod n$	($0 \le$ result $< n$)	RAM@0xE78	

27.4.5 Modular and Montgomery multiplication

To be more efficient when performing a sequence of multiplications, the PKA accelerates a multiplication that has at least one input in the Montgomery domain. The two main uses of this operation are:

- Map a value from natural domain to Montgomery domain and vice-versa
- Perform a modular multiplication $A \times B \bmod n$

The method to perform these operations is described next. The *x* function is this operation, and the *A*, *B*, and *C* operands are in the natural domain.

1. Inward or outward conversion to or from the Montgomery domain
   a) Assuming that *A* is an integer in the natural domain:
      – Compute *r2modn* using *Montgomery parameter computation*.
      – The result of $AR = A \times r2modn \bmod n$ is *A* in the Montgomery domain.
   b) Assuming that *BR* is an integer in the Montgomery domain:
      – The result of $B = BR \times 1 \bmod n$ is *B* in the natural domain.
      – Similarly, the *AR* value computed in *a)* can be converted into the natural domain by computing $A = AR \times 1 \bmod n$.
2. Simple modular multiplication $A \times B \bmod n$
   a) Compute *r2modn* using *Montgomery parameter computation*.
   b) Compute $AR = A \times r2modn \bmod n$. The output is in the Montgomery domain.
   c) Compute $AB = AR \times B \bmod n$. The output is in the natural domain.
3. Multiple modular multiplication $A \times B \times C \bmod n$
   a) Compute *r2modn* using *Montgomery parameter computation*.
   b) Compute $AR = A \times r2modn \bmod n$. The output is in the Montgomery domain.
   c) Compute $BR = B \times r2modn \bmod n$. The output is in the Montgomery domain.
   d) Compute $ABR = AR \times BR \bmod n$. The output is in the Montgomery domain.
   e) Compute $CR = C \times r2modn \bmod n$. The output is in the Montgomery domain.
   f) Compute $ABCR = ABR \times CR \bmod n$. The output is in the Montgomery domain.
   g) (optional) Repeat the two previous steps if multiplying with additional operands is required.
   h) Compute $ABC = ABCR \times 1 \bmod n$ to retrieve the result in the natural domain.

The following table summarizes the operation instructions.

Table 214. Montgomery multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x10	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < n$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < n$)	RAM@0xC68	
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
OUT	Result: $A \times B \bmod n$(1)	-	RAM@0xE78	

1. Result in Montgomery domain or in natural domain, depending upon the inputs nature (see examples *2* and *3*).

27.4.6 Modular exponentiation

Modular exponentiation operation is commonly used to perform a single-step RSA operation. It calculates of $A^e \bmod n$.

RSA operation involving public information (RSA encryption) can use the normal or fast mode detailed on *Table 215* and *Table 216*. RSA operation involving secret information (RSA decryption) must use the protected mode detailed on *Table 217*, for security reason.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware.

Note: When MODE[5:0] = 0x03, the error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

Operation instructions for modular exponentiation are summarized in *Table 215* (normal mode), *Table 216* (fast mode), and *Table 217* (protected mode). Fast mode usage is explained in *Section 27.3.6*.

Table 215. Modular exponentiation (normal mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x00	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	64 bits
	Operand length	(in bits, not null)	RAM@0x408	
IN/OUT	Operand *A* (base of exponentiation)	($0 \le A < n$)	RAM@0xC68	ROS
IN	Exponent *e*	($0 \le e < n$)	RAM@0xE78	
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
OUT	Result: A^e mod *n*	($0 \le$ result $< n$)	RAM@0x838	

Table 216. Modular exponentiation (fast mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x02	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	64 bits
	Operand length	(in bits, not null)	RAM@0x408	
IN/OUT	Operand *A* (base of exponentiation)	($0 \le A < n$)	RAM@0xC68	ROS
IN	Exponent *e*	($0 \le e < n$)	RAM@0xE78	
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
IN/OUT	Montgomery parameter R^2 mod *n*	(mandatory)	RAM@0x620	
OUT	Result: A^e mod *n*	($0 \le$ result $< n$)	RAM@0x838	

Table 217. Modular exponentiation (protected mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x03	PKA_CR	6 bits
	Exponent *e* length	(in bits, not null)	RAM@0x400	64 bits
	Modulus or operand length	(in bits, not null)	RAM@0x408	
	Operand *A* (base of exponentiation)	($0 \le A < n$)	RAM@0x16C8	ROS
	Exponent *e*	($0 \le e < n$)	RAM@0x14B8	
	Modulus value *n*	(odd integer only, $n < 2^{4096}$)	RAM@0x0838	
	Phi value(1)	-	RAM@0x0C68	
OUT	Result: A^e mod *n*	($0 \le$ result $< n$)	RAM@0x838	
ERROR	Error A^e mod *n*	No errors: 0xD60D Errors: 0xCBC9	RAM@0x1298	64 bits

1. Euler totient function of n^1 with phi = (p - 1) × (q - 1), where p and q are prime factors of modulus n (see NIST SP800-56B Appendix C or *RSA encryption/decryption principle* for details). As optimization it is recommended to keep phi information as part of the key pair generation. Alternative is to store the key as (p, q, e, d) instead of (N, e, d, *phi*), in this case, to derive N and *phi* using PKA arithmetic multiplier: $N = p \times q$, and *phi* = (p - 1) × (q - 1).

27.4.7 Modular inversion

The modular inversion operation calculates multiplicative inverse A^{-1} mod n. If the modulus n is prime, for all values of A ($1 \le A < n$) modular inversion output is valid. If the modulus n is not prime, A has an inverse only if the greatest common divisor between A and n is 1.

If the operand A is a divisor of the modulus n the result is a multiple of a factor of n.

The following table summarizes the operation instructions.

Table 218. Modular inversion

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x08	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < n$)	RAM@0xA50	ROS
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0xC68	
OUT	Result: A^{-1} mod *n*	0 < result < *n*	RAM@0xE78	

27.4.8 Modular reduction

The modular reduction operation calculates the remainder of A divided by n.

The following table summarizes the operation instructions.

Table 219. Modular reduction

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0D	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x400	64 bits
	Modulus length	(in bits, 8 < value < 4160)	RAM@0x408	
	Operand *A*	($0 \le A < 2n < 2^{4160}$)	RAM@0xA50	ROS
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0xC68	
OUT	Result *A* mod *n*	(0 < result < *n*)	RAM@0xE78	

27.4.9 Arithmetic addition

The arithmetic addition operation calculates *A* + *B*.

The following table summarizes the operation instructions.

Table 220. Arithmetic addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x09	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < 2^M$)	RAM@0xC68	
OUT	Result: *A* + *B*	($0 \le$ result $< 2^{M+1}$)	RAM@0xE78	ROS + 1

27.4.10 Arithmetic subtraction

The arithmetic subtraction operation calculates:

- ***A* - *B***, when $A \ge B$
- $\boldsymbol{A + 2^{int(M / 32) \times 32 + 1} - B}$, when $A < B$ and the residue of $M / 32$ is greater than zero
- $\boldsymbol{A + 2^{int(M / 32) \times 32} - B}$, when $A < B$ and the residue of $M / 32$ is zero

For the last two list items, the 32-bit word following the most significant word of the output is 0xFFFF FFFF, as the result is negative.

The following table summarizes the operation instructions.

Table 221. Arithmetic subtraction

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0A	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le < A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \le < B < 2^M$)	RAM@0xC68	
OUT	Result: *A* - *B*	($0 \le$ result $< 2^M$)	RAM@0xE78	

27.4.11 Arithmetic multiplication

The arithmetic multiplication operation calculates A × B.

The following table summarizes the operation instructions.

Table 222. Arithmetic multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0B	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < 2^M$)	RAM@0xC68	
OUT	Result: *A* × *B*	($0 \le$ result $< 2^{2M}$)	RAM@0xE78	2xROS

27.4.12 Arithmetic comparison

The arithmetic comparison operation yields the following results:

- **0xED2C**, when *A* = *B*
- **0x7AF8**, when *A* > *B*
- **0x916A**, when *A* < *B*

The following table summarizes the operation instructions.

Table 223. Arithmetic comparison

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0C	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < 2^M$)	RAM@0xC68	
OUT	If result *A* = *B*	0xED2C, 0x7AF8, or 0x916A	RAM@0xE78	64 bits

27.4.13 RSA CRT exponentiation

For efficiency many popular crypto libraries such as OpenSSL RSA use the following optimization for decryption and signing based on the chinese remainder theorem (CRT):

- *p* and *q* are precomputed primes, stored as part of the private key
- $d_P = d \bmod (p - 1)$
- $d_Q = d \bmod (q - 1)$ and
- $q_{inv} = q^{-1} \bmod p$

These values allow the recipient to compute the exponentiation m = A^d (mod pq) more efficiently as follows:

- $m_1 = A^{dP} \bmod p$
- $m_2 = A^{dQ} \bmod q$
- $h = q_{inv} (m_1 - m_2) \bmod p$, with $m_1 > m_2$
- $m = m_2 + hq \bmod pq$

Operation instructions for computing CRT exponentiation A^d mod pq are summarized in *Table 224*.

Table 224. CRT exponentiation

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x07	PKA_CR	6 bits
IN	Operand length	(in bits, not null)	RAM@0x408	64 bits
IN	Operand d_P	$(0 < d_P < 2^{M/2})$	RAM@0x730	ROS / 2
	Operand d_Q	$(0 < d_Q < 2^{M/2})$	RAM@0xE78	
	Operand q_{inv}	$(0 < q_{inv} < 2^{M/2})$	RAM@0x948	
	Prime p(1)	$(0 < p < 2^{M/2})$	RAM@0xB60	
	Prime q(1)	$(0 < q < 2^{M/2})$	RAM@0x1088	
IN	Operand A	$(0 < A < 2^M)$	RAM@0x12A0	ROS
OUT	Result: A^d mod pq	(0 ≤ result < pq)	RAM@0x838	

1. Must be different from 2.

27.4.14 Point on elliptic curve Fp check

This operation checks whether a given point P (x, y) satisfies or not the curves over prime fields equation $y^2 = (x^3 + ax + b) \bmod p$, where a and b are elements of the curve.

The following table summarizes the operation instructions.

Table 225. Point on elliptic curve *Fp* check

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x28	PKA_CR	6 bits
	Modulus length	(in bits, not null, 8 < value < 640)	RAM@0x408	64 bits
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	
	Curve coefficient |*a*|	(absolute value, |*a*| < *p*)	RAM@0x418	EOS
	Curve coefficient *b*	(|*b*| < *p*)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, 0 < *p* < 2640)	RAM@0x470	
	Point P coordinate *x*	(*x* < p)	RAM@0x578	
	Point P coordinate *y*	(*y* < *p*)	RAM@0x5D0	
	Montgomery parameter R^2 mod *n*	-	RAM@0x4C8	
OUT	Result: point *P* on curve	0xD60D: point on curve 0xA3B7: point not on curve 0xF946: *x* or *y* coordinate is not smaller than modulus *p*	RAM@0x680	64 bits

27.4.15 ECC Fp scalar multiplication

This operation calculates $k \times P\,(x_P, y_P)$, where *P* is a point on a curve over prime fields and *x* is the elliptic curve scalar point multiplication. The result is a point that belongs to the same curve or a point at infinity.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events, a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware. PKA tamper is also triggered when the programmed input point is not found on the input ECC curve. PKA operation *point on elliptic curve* can be used to avoid this.

The following table summarizes the operation instructions.

Table 226. ECC Fp scalar multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x20	PKA_CR	6 bits
IN	Curve prime order *n* length	(in bits, not null,)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	

Table 226. ECC Fp scalar multiplication (continued)

Parameters with direction		Value (note)	Storage	Size
IN	Curve coefficient |*a*|	(absolute value, |*a*| < *p*)	RAM@0x418	EOS
	Curve coefficient *b*	(positive integer)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x1088	
	Scalar multiplier *k*	($0 \le k < n$)	RAM@0x12A0	
	Point P coordinate x_P	($x_P < p$)	RAM@0x578	
	Point P coordinate y_P	($y_P < p$)	RAM@0x470	
	Curve prime order *n*	(integer prime)	RAM@0xF88	
OUT	Result: *k* × *P* coordinate *x'*	(result < *p*)	RAM@0x578	EOS
	Result: *k* × *P* coordinate *y'*	(result < *p*)	RAM@0x5D0	
ERROR	Error *k* × *P*	No errors: 0xD60D Errors: 0xCBC9	RAM@0x680	64 bits

When performing this operation the following special cases must be noted:

- For $k = 0$ this function returns a point at infinity (0, 0) if curve parameter *b* is nonzero, (0, 1) otherwise. For *k* different from 0 it might happen that a point at infinity is returned. When the application detects this behavior, a new computation must be carried out.
- For $k < 0$ (that is, a negative scalar multiplication is required) the multiplier absolute value
  $k = |-k|$ must be provided to the PKA. After the computation completion, the formula $-P = (x, -y)$ can be used to compute the *y* coordinate of the effective final result (the *x* coordinate remains the same).

Note: The error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

27.4.16 ECDSA sign

Table 227 (input parameters) and *Table 228* (output parameters) summarize the ECDSA signing operation.

Upon an output error different from 0xD60D, the application must generate a new *k* and repeat the ECDSA sign operation.

Note: As long as this operation is ongoing (BUSY = 1), the PKA control register and the PKA memory are not accessible. The access is restored when the PKA clears the BUSY bit.

The operation completion with errors due to unexpected hardware events triggers a PKA tamper event in the TAMP peripheral and blocks access to the PKA RAM until the hardware erases the event. Missing programmed input point on the input ECC curve also triggers a PKA tamper event. The use of the *point on elliptic curve* PKA operation prevents this.

Table 227. ECDSA sign - Inputs

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x24	PKA_CR	6 bits
	Curve prime order *n* length (*nlen*)	(in bits, not null)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	
	Curve coefficient |*a*|	(absolute value, |*a*| < *p*)	RAM@0x418	EOS
	Curve coefficient *b*	(positive integer)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x1088	
	Integer *k*(1)	($0 \leq k <$ n)	RAM@0x12A0	
	Curve base point *G* coordinate *x*	($x < p$)	RAM@0x578	
	Curve base point *G* coordinate *y*	($y < p$)	RAM@0x470	
	Hash of message *z*	(hash size equal to *nlen*)(2)	RAM@0xFE8	
	Private key *d*	($0 < d$)	RAM@0xF28	
	Curve prime order *n*	(integer prime)	RAM@0xF88	

1. This integer is usually a cryptographically secure random number, but in some cases *k* can be deterministically generated.
2. To match the hash parameter size to the length of the curve prime order *n,* pad the result with zeros or use the hash truncation.

Table 228. ECDSA sign - Outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Signature part *r*	($0 < r < n$)	RAM@0x730	EOS
	Signature part *s*	($0 < s < n$)	RAM@0x788	
ERROR	Result of signature	0xD60D: successful computation, no error 0xCBC9: failed computation 0xA3B7: signature part r is equal to 0 0xF946: signature part s is equal to 0	RAM@0xFE0	64 bits

Note: The error code is not affected when the PKA automatically clears the PKA RAM at the end of this protected operation. The error output 0xD60D does not affect the result output either.

Extended ECDSA support

The PKA also supports the extended ECDSA signature, which accepts the same inputs as the standard ECDSA signature (see *Table 227*) and produces similar outputs (see *Table 228*), with the addition of the point *kG* coordinates. The following table defines these additional outputs.

Table 229. Extended ECDSA sign - additional outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Curve point kG coordinate x_1	($0 \le x_1 < p$)	RAM@0x1400	EOS
	Curve point kG coordinate y_1	($0 \le y_1 < p$)	RAM@0x1458	

27.4.17 ECDSA verification

Table 230 (input parameters) and *Table 231* (output parameters) summarize the ECDSA verification operation.

Upon an output error different from 0xD60D, the signature is not verified.

Table 230. ECDSA verification - inputs

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x26	PKA_CR	6 bits
	Curve prime order n length (*nlen*)	(in bits, not null)	RAM@0x408	64 bits
	Curve modulus p length	(in bits, not null, 8 < value < 640)	RAM@0x4C8	
	Curve coefficient a sign	0x0: positive 0x1: negative	RAM@0x468	
	Curve coefficient $\|a\|$	(absolute value, $\|a\| < p$)	RAM@0x470	EOS
	Curve modulus value p	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x4D0	
	Curve base point G coordinate x	($x < p$)	RAM@0x678	
	Curve base point G coordinate y	($y < p$)	RAM@0x6D0	
	Public-key curve point Q coordinate x_Q	($x_Q < p$)	RAM@0x12F8	
	Public-key curve point Q coordinate y_Q	($y_Q < p$)	RAM@0x1350	
	Signature part r	($0 < r < n$)	RAM@0x10E0	
	Signature part s	($0 < s < n$)	RAM@0xC68	
	Hash of message z	(hash size equal to *nlen*)(1)	RAM@0x13A8	
	Curve prime order n	(integer prime)	RAM@0x1088	

1. To match the hash parameter size to the length of the curve prime order n, pad the result with zeros or use the hash truncation.

Table 231. ECDSA verification - outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Result: ECDSA verify	– 0xD60D: valid signature – 0xA3B7: invalid signature	RAM@0x5D0	64 bits
	Computed signature part *r*	– ($0 < r < n$)	RAM@0x578	EOS

27.4.18 ECC complete addition

The ECC complete addition operation calculates the sum of two points on an elliptic curve.

Operation instructions are summarized in *Table 232*.

Note: The two input points and the resulting point are represented in Jacobian coordinates (X,Y,Z). To input a point in affine coordinates (x, y) conversion (X, Y, Z) = (x, y, 1) can be used. To convert resulting point to Jacobian coordinates conversion (x, y)= (X / Z^2, Y / Z^3) can be used.

Table 232. ECC complete addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x23	PKA_CR	6 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	64 bits
	Curve coefficient *a* sign	– 0x0: positive 0x1: negative	RAM@0x410	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	EOS
	Curve coefficient $\|a\|$	(absolute value, $\|a\| < p$)	RAM@0x418	
	First point P coordinate *X*	($x < p$)	RAM@0x628	
	First point P coordinate *Y*	($y < p$)	RAM@0x680	
	First point P coordinate *Z*	($z < p$)	RAM@0x6D8	
	Second point Q coordinate *X*	($x < p$)	RAM@0x730	
	Second point Q coordinate *Y*	($y < p$)	RAM@0x788	
	Second point Q coordinate *Z*	($z < p$)	RAM@0x7E0	
OUT	Result coordinate *X*	($x < p$)	RAM@0xD60	
	Result coordinate *Y*	($y < p$)	RAM@0xDB8	
	Result coordinate *Z*	($z < p$)	RAM@0xE10	

27.4.19 ECC double base ladder

ECC double base ladder operation consists in the computation of $k \times P + m \times Q$, where (P, Q) are two points on an elliptic curve and (k, m) are two scalars. Operation instructions are summarized in *Table 233*.

If the resulting point is the point at infinity (error code 0xA3B7), the resulting coordinate equals to (0, 0).

Note: The two input points are represented in Jacobian coordinates (X, Y , Z). To input a point in affine coordinates (x, y) conversion (X, Y, Z) = (x, y, 1) can be used. The result is represented in affine coordinates (x ,y)

This operation requires the input point Z coordinate to be equal to 1 (input point represented in affine coordinates).

Table 233. ECC double base ladder

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x27	PKA_CR	6 bits
	Curve prime order *n* length	(in bits, not null)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	– 0x0: positive – 0x1: negative	RAM@0x410	EOS
	Curve coefficient |*a*|	(absolute value, |a| < p)	RAM@0x418	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	
	Integer *k*	($0 < k < 2^{640}$)	RAM@0x520	
	Integer *m*	($0 < m < 2^{640}$)	RAM@0x578	
	First point *P* coordinate *X*	($x < p$)	RAM@0x628	
	First point *P* coordinate *Y*	($y < p$)	RAM@0x680	
	First point *P* coordinate *Z*	($z < p$)	RAM@0x6D8	
	Second point *Q* coordinate *X*	($x < p$)	RAM@0x730	
	Second point *Q* coordinate *Y*	($y < p$)	RAM@0x788	
	Second point *Q* coordinate *Z*	($z < p$)	RAM@0x7E0	
OUT	Result coordinate *x*	($x < p$)	RAM@0x578	EOS
	Result coordinate *y*	(y < *p*)	RAM@0x5D0	
	Error code	– Point not at infinity: 0xD60D – Point at infinity: 0xA3B7	RAM@0x520	64 bits

27.4.20 ECC projective to affine

ECC projective to affine operation calculates the conversion between the representation of a point P in homogeneous projective coordinates and the representation of the point P in affine coordinates. Namely, if the point is represented by the triple (*X*, *Y*, *Z*), it calculates the affine coordinates (*x*, *y*) = (*X* / *Z*, *Y* / *Z*).

All the operations are performed modulo the modulus p of the curve, which the point belongs to. If the resulting point is the point at infinity (error code 0xA3B7), resulting coordinate equals (0,0).

Operation instructions are summarized in *Table 234*.

Table 234. ECC projective to affine

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x2F	PKA_CR	6 bits
	Curve modulus *p* length	(in bits, 8 < value < 640)	RAM@0x408	64 bits
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	EOS
	Point P coordinate *X* (projective)	($x < p$)	RAM@0xD60	
	Point P coordinate *Y* (projective)	($y < p$)	RAM@0xDB8	
	Point P coordinate *Z* (projective)	($z < p$)	RAM@0xE10	
	Montgomery parameter R^2 mod *n*	-	RAM@0x4C8	
OUT	Point P coordinate *x* (affine)	($x < p$)	RAM@0x578	
	Point P coordinate *y* (affine)	($y < p$)	RAM@0x5D0	
ERROR	Error code	Point not at infinity: 0xD60D Point at infinity: 0xA3B7	RAM@0x680	64 bits

27.5 Example of configurations and processing times

27.5.1 Supported elliptic curves

The PKA supports all non-singular elliptic curves defined over prime fields. Those curves can be described with a short Weierstrass equation, $y^2 = x^3 + ax + b$ (mod p).

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.
The maximum supported operand size for ECC operations is 640 bits.

When publishing the ECC domain parameters of those elliptic curves, standard bodies define the following parameters:

- the prime integer *p*, used as the modulus for all point arithmetic in the finite field *GF(p)*
- the (usually prime) integer *n*, the order of the group generated by *G*, defined below
- the base point of the curve *G*, defined by its coordinates (*Gx*, *Gy*)
- the integers *a* and *b*, coefficients of the short Weierstrass equation.

For the last bullet, when standard bodies define *a* as negative, PKA supports two representations:

1. ***a* defined as *p*-|*a*|** in the finite field *GF(p)*, for example ***p*-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**F**
   Curve coefficient a sign= 0x0 (positive)
   Curve coefficient a = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**C**
2. ***a* defined as negative**, for example **-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF
   Curve coefficient a sign= 0x**1** (negative)
   Curve coefficient a = 0x00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000000**3**

Table 235 summarizes the family of curves supported by PKA for ECC operations.

Table 235. Family of supported curves for ECC operations

Curve name	Standard	Reference	
P-192	NIST	*Digital Signature Standard (DSS)*, NIST FIPS 186-4	
P-224			
P-256			
P-384			
P-521			
brainpoolP224r1, brainpoolP224t1	IETF	– *Brainpool Elliptic Curves*, IETF RFC 5639 – *Brainpool Elliptic Curves for the Internet Key Exchange (IKE) Group Description Registry*, IETF RFC 6932	https://tools.ietf.org
brainpoolP256r1, brainpoolP256t1			
brainpoolP320r1, brainpoolP320t1			
brainpoolP384r1, brainpoolP384t1			
brainpoolP512r1, brainpoolP512t1			
secp192k1, secp192r1	SEC	*Standards for Efficient Cryptography SEC 2 curves*	https://www.secg.org
secp224k1, secp224r1			
secp256k1, secp256r1			
secp384r1			
secp521r1			
Recommended curve parameters for public key cryptographic algorithm SM2	OSCCA	– *Public key cryptographic algorithm SM2 based on elliptic curves*, Organization of State Commercial Administration of China OSCCA SM2, December 2010 – *Digital signatures - Part 3 Discrete logarithm based mechanisms*, ISO/IEC 14888-3, November 2018	

27.5.2 Computation times

The following tables summarize the PKA computation times, expressed in AHB clock cycles.

Table 236. Modular exponentiation

Exponent length (in bits)	Mode	Modulus length (in bits)			
		1024	2048	3072	4096
3	Normal	124600	491000	684000	1133200
	Fast	22700	82000	178000	311000
17	Normal	135700	531400	772400	1288000
	Fast	33800	122500	266500	465800
2^{16} + 1	Normal	180000	693700	1126200	1907200
	Fast	78200	284700	620400	1085000
1024	Protected	9958000	-	-	-
	Normal	5850000	-	-	-
	Fast	5748000	-	-	-
	CRT(1)	1775000	-	-	-
2048	Protected	-	63886000	-	-
	Normal	-	42240000	-	-
	Fast	-	41832000	-	-
	CRT(1)	-	11670000	-	-
3072	Protected	-	-	199403000	-
	Normal	-	-	136830000	-
	Fast	-	-	136325000	-
	CRT(1)	-	-	36886000	-
4096	Protected	-	-	-	454318000
	Normal	-	-	-	316000000
	Fast	-	-	-	315226000
	CRT(1)	-	-	-	84577000

1. CRT stands for chinese remainder theorem optimization (MODE[5:0] bitfield= 0x07).

Table 237. ECC scalar multiplication(1)

Modulus length (in bits)							
160	192	256	320	384	512	521	640
-	1590000	3083000	5339000	8518000	17818000	21053000	31826000

1. These times depend on the number of 1s included in the scalar parameter, and include the computation of Montgomery parameter R2.

Table 238. ECDSA signature average computation time(1) (2)

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
-	1500000	2744000	4579000	7184000	14455000	16685000	24965000

1. These values are average execution times of random moduli of given length, as they depend upon the length and the value of the modulus.
2. The execution time for the moduli that define the finite field of NIST elliptic curves is shorter than that needed for the moduli used for Brainpool elliptic curves or for random moduli of the same size.

Table 239. ECDSA verification average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
1011000	1495000	2938000	5014000	7979000	16804000	19254000	29582000

Table 240. ECC double base ladder average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
967000	1419000	2768000	4784000	7547000	15854000	18257000	28257000

Table 241. ECC projective to affine average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**640**
47600	78000	148300	253000	419000	838400	1049300

Table 242. ECC complete addition average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**640**
10000	12000	18000	26000	39000	63000	89000

Table 243. Point on elliptic curve Fp check average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
3400	4200	6100	8300	10900	17200	-	-

Table 244. Montgomery parameters average computation times(1)

Modulus length (in bits)							
192	**256**	**320**	**512**	**1024**	**2048**	**3072**	**4096**
8600	8710	11870	17000	102000	410000	506000	822000

1. The computation times depend upon the length and the value of the modulus, hence these values are average execution times of random moduli of given length.

27.6 PKA in low-power modes

Table 245. Effect of low-power modes on PKA

Mode	Description
Sleep	No effect.
Stop 0 and Stop 1	The PKA register contents are kept.
Stop 2 and Stop 3	The PKA peripheral is powered down. It must be reinitialized upon existing this low-power mode.
Standby	

27.7 PKA interrupts

There are four individual maskable interrupt sources generated by the public key accelerator, signaling the following events:

1. PKA unsupported operation error (OPERRF), see *Section 27.3.7*
2. Access to unmapped address (ADDRERRF), see *Section 27.3.7*
3. PKA RAM access while PKA operation is in progress (RAMERRF), see *Section 27.3.7*
4. PKA end of operation (PROCENDF)

The interrupt sources are connected to the same global interrupt request signal pka_it.

The user can enable or disable above interrupt sources individually by changing the mask bits in the *PKA control register (PKA_CR)*. Setting the appropriate mask bit to 1 enables the interrupt. The status of the individual interrupt events can be read from the PKA status register (PKA_SR), and it is cleared in PKA_CLRFR register.

Table 246 gives a summary of the available features.

Table 246. PKA interrupt requests

Acronym	Event	Event flag	Enable control bit	Clear method
PKA	Unsupported operation	OPERRF	OPERRIE	Set OPERRFC bit
	Access to unmapped address error	ADDRERRF	ADDRERRIE	Set ADDRERRFC bit
	PKA RAM access error	RAMERRF	RAMERRIE	Set RAMERRFC bit
	PKA end of operation	PROCENDF	PROCENDIE	Set PROCENDFC bit

27.8 PKA registers

27.8.1 PKA control register (PKA_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRIE	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.
										rw	rw	rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MODE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	START	EN
		rw	rw	rw	rw	rw	rw							rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **OPERRIE**: Operation error interrupt enable
0: No interrupt is generated when OPERRF flag is set in PKA_SR.
1: An interrupt is generated when OPERRF flag is set in PKA_SR.

Bit 20 **ADDRERRIE**: Address error interrupt enable
0: No interrupt is generated when ADDRERRF flag is set in PKA_SR.
1: An interrupt is generated when ADDRERRF flag is set in PKA_SR.

Bit 19 **RAMERRIE**: RAM error interrupt enable
0: No interrupt is generated when RAMERRF flag is set in PKA_SR.
1: An interrupt is generated when RAMERRF flag is set in PKA_SR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDIE**: End of operation interrupt enable
0: No interrupt is generated when PROCENDF flag is set in PKA_SR.
1: An interrupt is generated when PROCENDF flag is set in PKA_SR.

Bits 16:14 Reserved, must be kept at reset value.

Bits 13:8 **MODE[5:0]**: PKA operation code

000000: Montgomery parameter computation then modular exponentiation
000001: Montgomery parameter computation only
000010: Modular exponentiation only (Montgomery parameter must be loaded first)
000011: Modular exponentiation (protected, used when manipulating secrets)
100000: Montgomery parameter computation then ECC scalar multiplication (protected)
100100: ECDSA sign (protected)
100110: ECDSA verification
101000: Point on elliptic curve Fp check
000111: RSA CRT exponentiation
001000: Modular inversion
001001: Arithmetic addition
001010: Arithmetic subtraction
001011: Arithmetic multiplication
001100: Arithmetic comparison
001101: Modular reduction
001110: Modular addition
001111: Modular subtraction
010000: Montgomery multiplication
100011: ECC complete addition
100111: ECC double base ladder
101111: ECC projective to affine

When an operation not listed here is written by the application with EN bit set, OPERRF bit is set in PKA_SR register, and the write to MODE[5:0] bitfield is ignored. When PKA is configured in limited mode (LMF = 1 in PKA_SR), writing a MODE[5:0] different from 0x26 with EN bit to 1 triggers OPERRF bit to be set and write to MODE[5:0] bit is ignored.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **START**: start the operation

Set this bit to start the operation selected by the MODE[5:0] bitfield, using the operands and data already written to the PKA RAM. This bit is always read as 0.

When an illegal operation is selected while START bit is set no operation is started, and OPERRF bit is set in PKA_SR.

Note: START is ignored if PKA is busy.

Bit 0 **EN**: PKA enable

0: Disable PKA

1: Enable PKA; PKA becomes functional when INITOK is set by hardware in PKA_SR. When an illegal operation is selected while EN = 1, OPERRF bit is set in PKA_SR. See PKA_CR.MODE[5:0] bitfield for details.

Note: When EN = 0, PKA RAM can still be accessed by the application.

27.8.2 PKA status register (PKA_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRF	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY
										r	r	r		r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMF	INITOK
														r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **OPERRF**: Operation error flag

0: No event error

1: An illegal or unknown operation has been selected in PKA_CR register

This bit is cleared using OPERRFC bit in PKA_CLRFR.

Bit 20 **ADDRERRF**: Address error flag

0: No address error

1: Address access is out of range (unmapped address)

This bit is cleared using ADDRERRFC bit in PKA_CLRFR.

Bit 19 **RAMERRF**: PKA RAM error flag

0: No PKA RAM access error

1: An AHB access to the PKA RAM occurred while the PKA core was computing and using its internal RAM (AHB PKA_RAM access are not allowed while PKA operation is in progress).

This bit is cleared using RAMERRFC bit in PKA_CLRFR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDF**: PKA end of operation flag

0: Operation in progress

1: PKA operation is completed. This flag is set when the BUSY bit is deasserted.

Bit 16 **BUSY**: Busy flag

This bit is set whenever a PKA operation is in progress (START = 1 in PKA_CR). It is automatically cleared when the computation is complete, making PKA RAM accessible again.

0: No operation is in progress (default)

1: An operation is in progress

If PKA is started with a wrong opcode, it stays busy for a couple of cycles, then it aborts automatically the operation and goes back to ready (BUSY = 0).

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **LMF**: Limited mode flag

This bit is updated when EN bit in PKA_CR is set

0: All values documented in MODE[5:0] bitfield can be used.

1: Only ECDSA verification (MODE[5:0] = 0x26) is supported by the PKA.

Bit 0 **INITOK**: PKA initialization OK
This bit is asserted when PKA initialization is complete.
0: PKA is not initialized correctly. START bit cannot be set.
1: PKA is initialized correctly and can be used normally.
When RNG is not able to output proper random numbers or when SAES is using the RNG bus, INITOK remains at 0. If it happens, make sure the RNG peripheral is properly clocked and initialized, and the SAES peripheral is clocked with BUSY bit cleared in SAES_SR to release the RNG bus, then enable PKA again.

27.8.3 PKA clear flag register (PKA_CLRFR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRFC	ADDRERRFC	RAMERRFC	Res.	PROCENDFC	Res.
										w	w	w		w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **OPERRFC**: Clear operation error flag
0: No action
1: Clear the OPERRF flag in PKA_SR

Bit 20 **ADDRERRFC**: Clear address error flag
0: No action
1: Clear the ADDRERRF flag in PKA_SR

Bit 19 **RAMERRFC**: Clear PKA RAM error flag
0: No action
1: Clear the RAMERRF flag in PKA_SR

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDFC**: Clear PKA end of operation flag
0: No action
1: Clear the PROCENDF flag in PKA_SR

Bits 16:0 Reserved, must be kept at reset value.

Note: Reading PKA_CLRFR returns all 0s.

27.8.4 PKA RAM

The PKA RAM is mapped at the offset address of 0x0400 compared to the PKA base address. Only 32-bit word single accesses are supported, through PKA.AHB interface.

RAM size is 5336 bytes (max word offset: 0x14D0)

Note: PKA RAM cannot be used just after a PKA reset or a product reset, as described in Section 27.3.3: PKA reset and clocks.

27.8.5 PKA register map

Table 247. PKA register map and reset values

Offset	Register name	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
0x000	PKA_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRIE	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.	Res.	Res.	MODE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	START	EN
	Reset value											0	0	0		0				0	0	0	0	0	0							0	0
0x004	PKA_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRF	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMF	INITOK
	Reset value											0	0	0		0	0															0	0
0x008	PKA_CLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRFC	ADDRERRFC	RAMERRFC	Res.	PROCENDFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0	0	0		0																	

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

28 General-purpose timer (TIM2)

In this section, “TIMx” should be understood as “TIM2” since there is only one instance of this type of timer for the products to which this reference manual applies.

28.1 TIM2 introduction

The general-purpose timers consist of a 16-bit or 32-bit autoreload counter driven by a programmable prescaler.

They can be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 28.4.23: Timer synchronization*.

28.2 TIM2 main features

General-purpose TIMx timer features include:

- 16-bit or 32-bit up, down, up/down autoreload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to four independent channels for:
  - Input capture.
  - Output compare.
  - PWM generation (edge- and center-aligned modes).
  - One-pulse mode output.
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger).
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger).
  - Input capture.
  - Output compare.
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes.
- Trigger input for external clock or cycle-by-cycle current management.
- ADC synchronization for jitter-free sampling points.

28.3 TIM2 implementation

Table 248. General purpose timers(1)

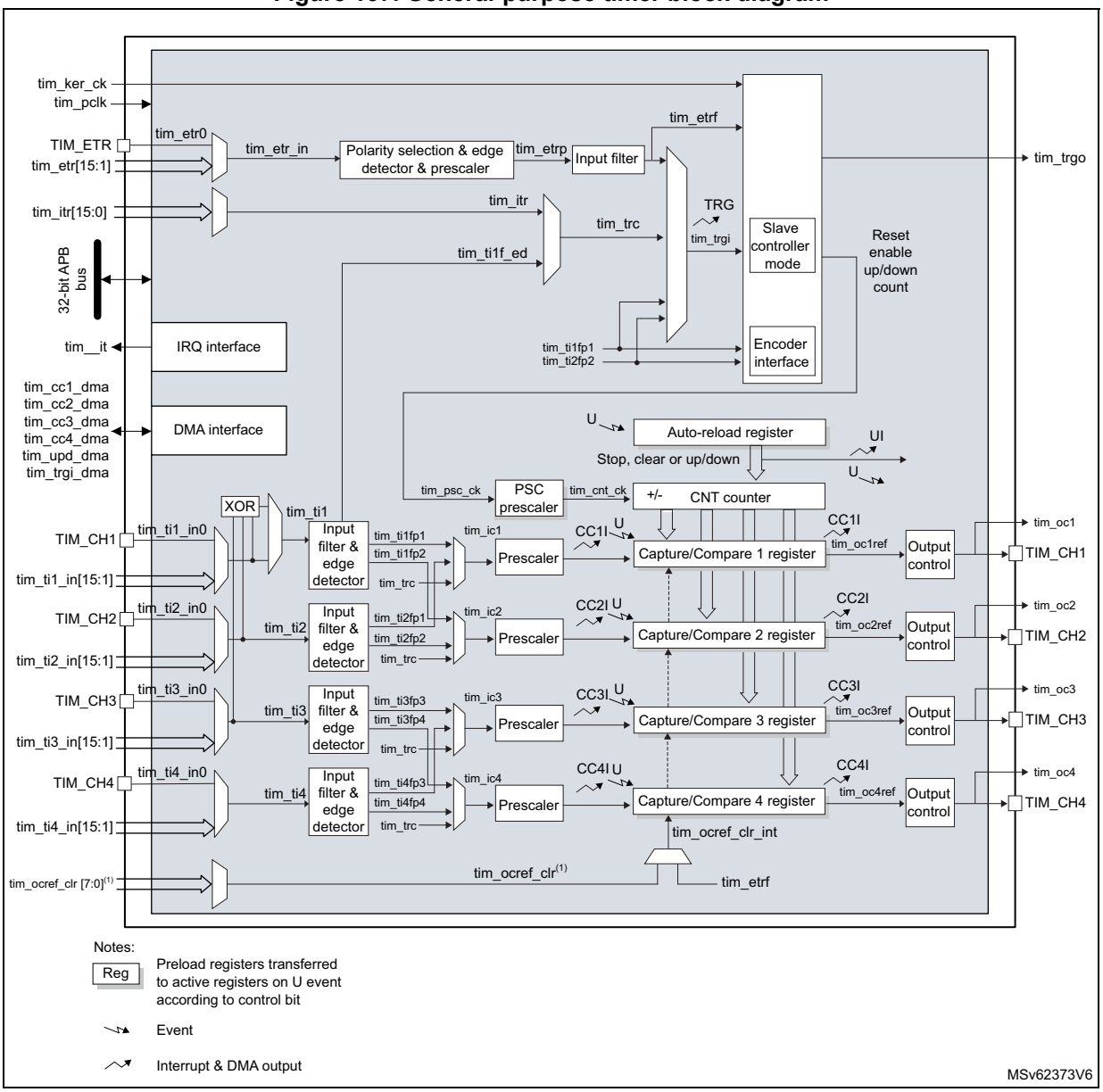
Timer instance	TIM2
CHx	1 to 4
ETR	X
Resolution	32-bit
DMA request channel	1 to 4
DMA request update	Yes
DMA request trigger	No
OCREF clear sources	tim2_etrf tim2_ocref_clr[7:0]

1. Note: ‘X’ = supported, ‘-’= not supported.

28.4 TIM2 functional description

28.4.1 Block diagram

Figure 137. General-purpose timer block diagram



1. This feature is not available on all timers, refer to *Section 28.3: TIM2 implementation*.

28.4.2 TIM2 pins and internal signals

Table 249 and *Table 250* in this section summarize the TIM inputs and outputs.

Table 249. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) , external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.

Table 250. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[15:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulse width control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[15:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulse width control.
tim_pclk	Input	Timer APB clock.
tim_ker_ck	Input	Timer kernel clock

Table 250. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_it	Output	Global Timer interrupt, gathering capture/compare, update and break trigger requests.
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture/compare [4:1] dma requests.
tim_upd_dma	Output	Timer update dma request.
tim_trgi_dma	Output	Timer trigger dma request.

Table 251, *Table 252*, *Table 253* and *Table 254* are listing the sources connected to the tim_ti[4:1] input multiplexers.

Table 251. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources
	TIM2
tim_ti1_in0	TIM2_CH1
tim_ti1_in[15:1]	Reserved

Table 252. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources
	TIM2
tim_ti2_in0	TIM2_CH2
tim_ti2_in[15:1]	Reserved

Table 253. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources
	TIM2
tim_ti3_in0	TIM2_CH3
tim_ti3_in[15:1]	Reserved

Table 254. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources
	TIM2
tim_ti4_in0	TIM2_CH4
tim_ti4_in[15:1]	Reserved

Table 255 lists the internal sources connected to the tim_etr input multiplexer.

Table 255. TIMx internal trigger connection

TIMx	TIM2
tim_itr[6:0]	Reserved
tim_itr7	tim16_oc1
tim_itr8	tim17_oc1
tim_itr[15:9]	Reserved

Table 256 lists the internal sources connected to the tim_etr input multiplexer.

Table 256. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signals assignment
	TIM2
tim_etr0	TIM2_ETR
tim_etr[3:1]	Reserved
tim_etr4	HSI16
tim_etr[10:5]	Reserved
tim_etr11	LSE
tim_etr12	adc4_awd1
tim_etr13	adc4_awd2
tim_etr14	adc4_awd3
tim_etr15	Reserved

Table 257 lists the internal sources connected to the tim_ocref_clr input multiplexer.

Table 257. Interconnect to the tim_ocref_clr input multiplexer

Timer tim_ocref_clr signal	Timer tim_ocref_clr signals assignment
	TIM2
tim_ocref_clr[7:0]	Reserved

28.4.3 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related autoreload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR).

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when down-counting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set one clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 138 and *Figure 139* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 138. Counter timing diagram with prescaler division change from 1 to 2

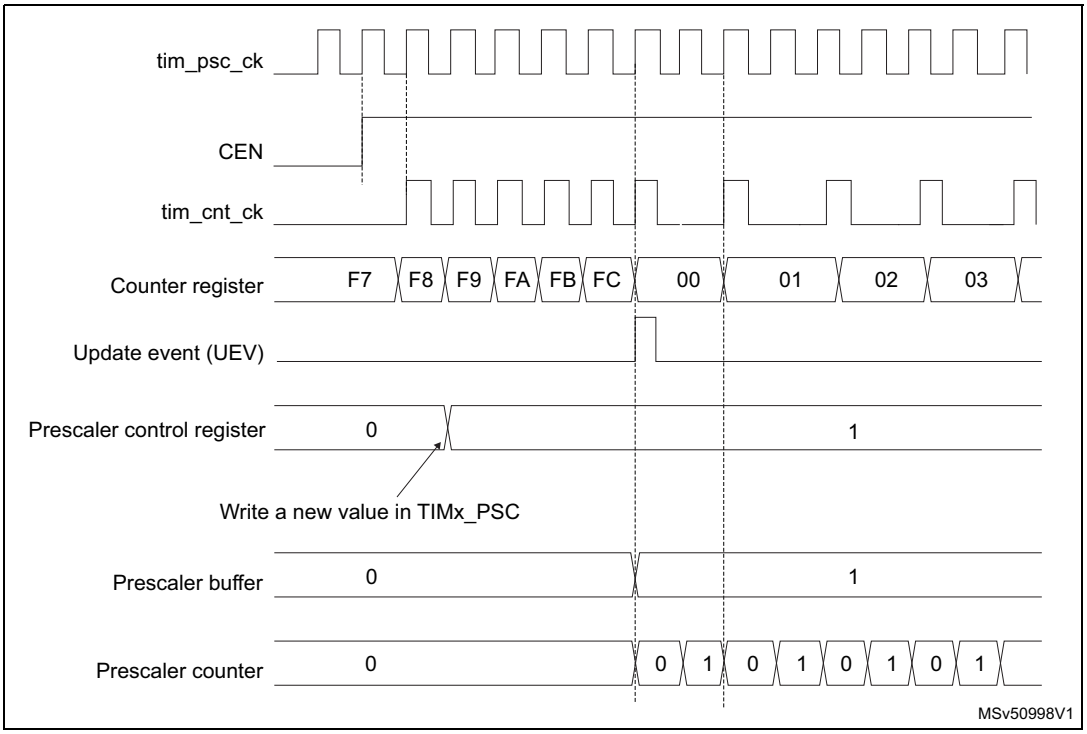
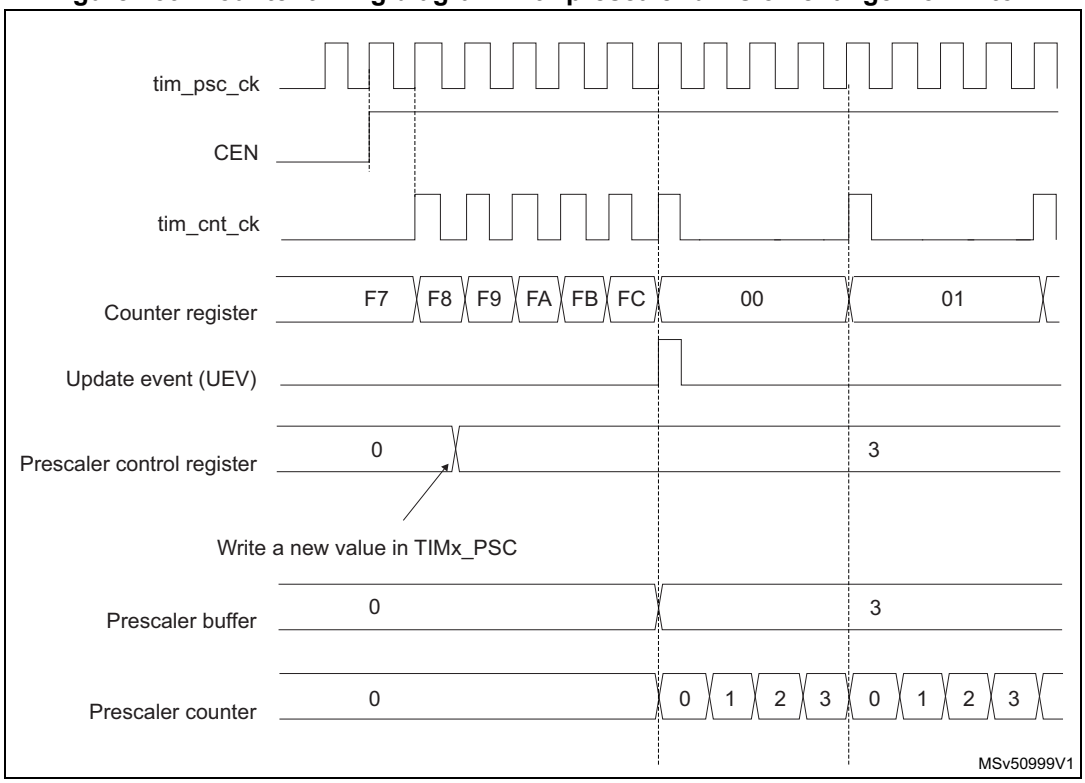


Figure 139. Counter timing diagram with prescaler division change from 1 to 4



28.4.4 Counter modes

Up-counting mode

In up-counting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload shadow register is updated with the preload value (TIMx_ARR).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 140. Counter timing diagram, internal clock divided by 1

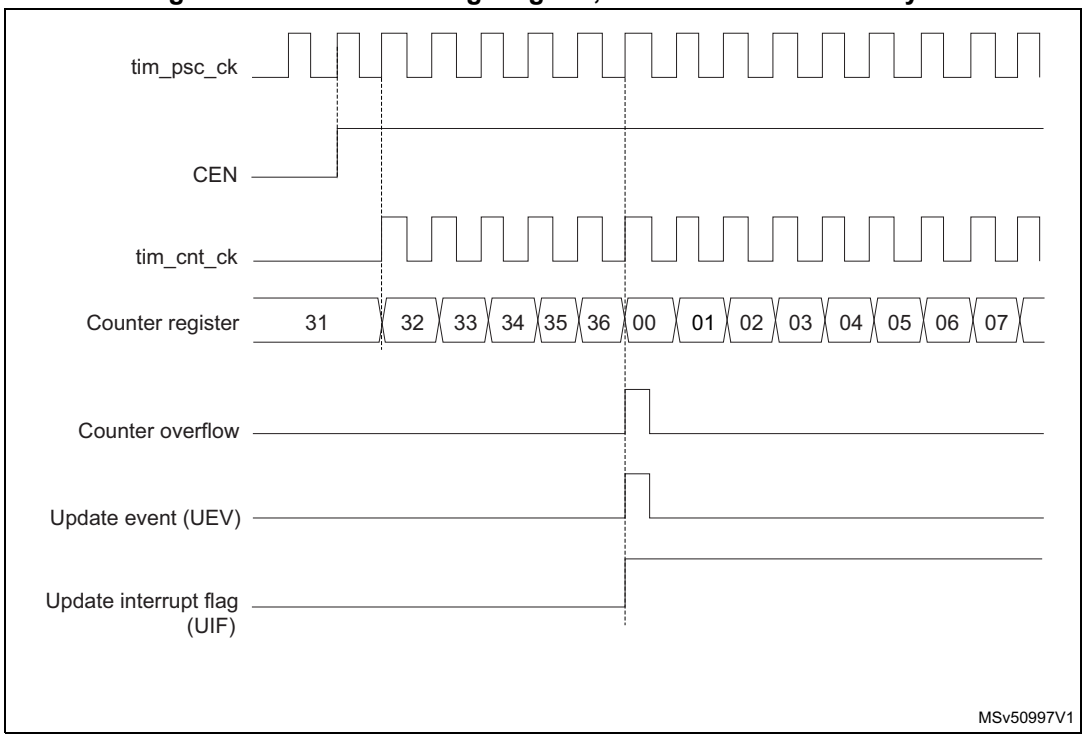


Figure 141. Counter timing diagram, internal clock divided by 2

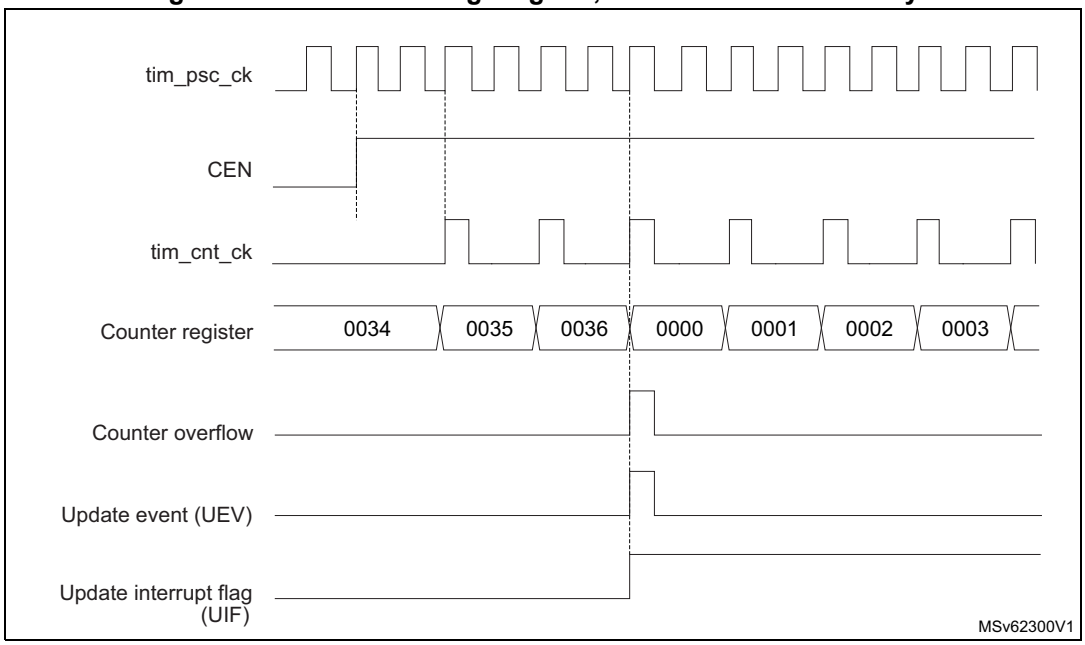


Figure 142. Counter timing diagram, internal clock divided by 4

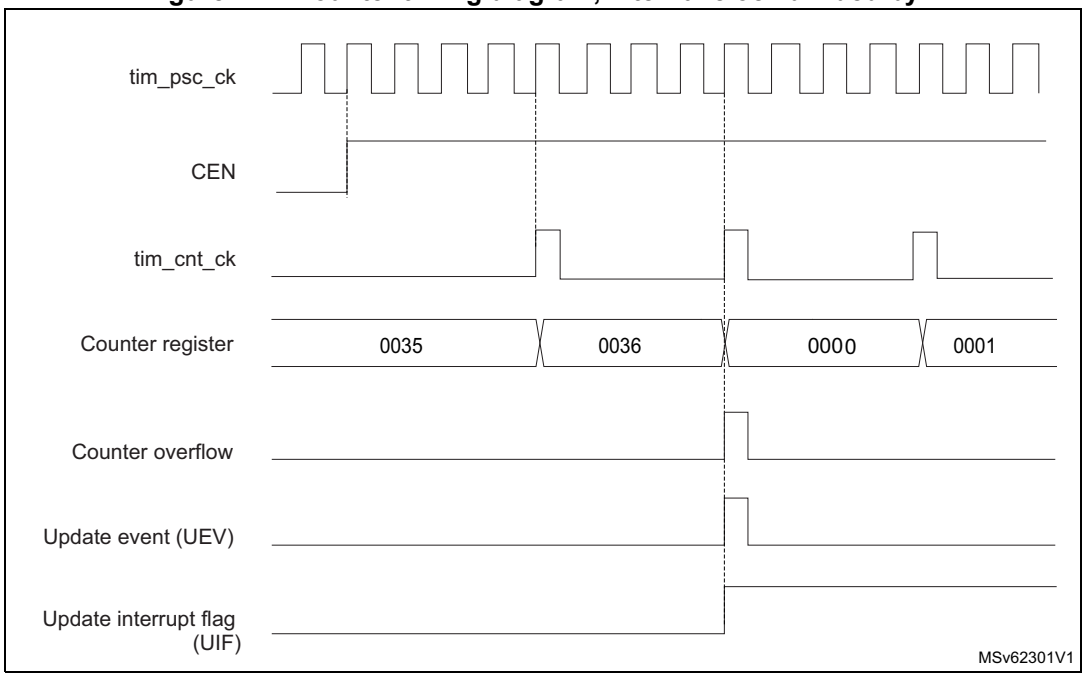


Figure 143. Counter timing diagram, internal clock divided by N

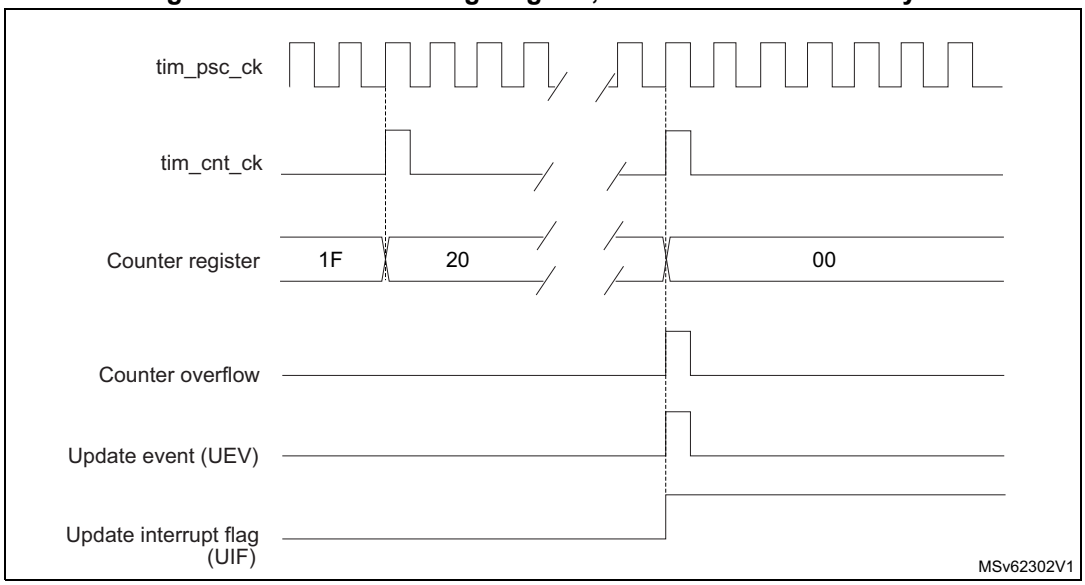


Figure 144. Counter timing diagram, Update event when ARPE = 0 (TIMx_ARR not preloaded)

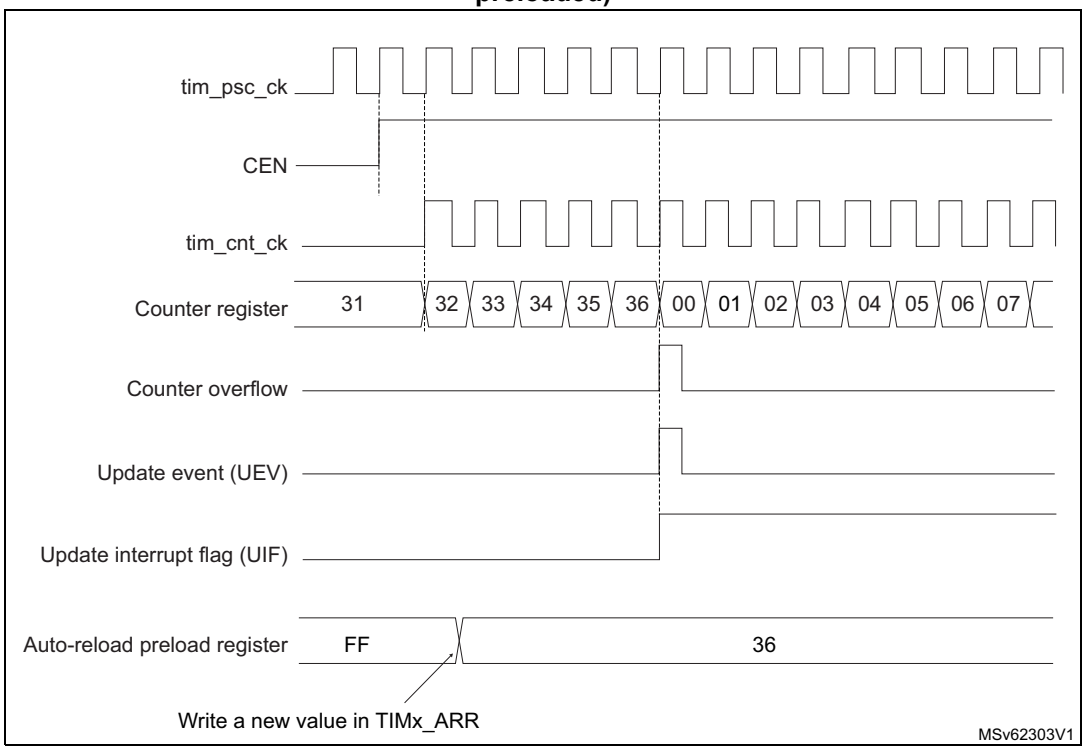
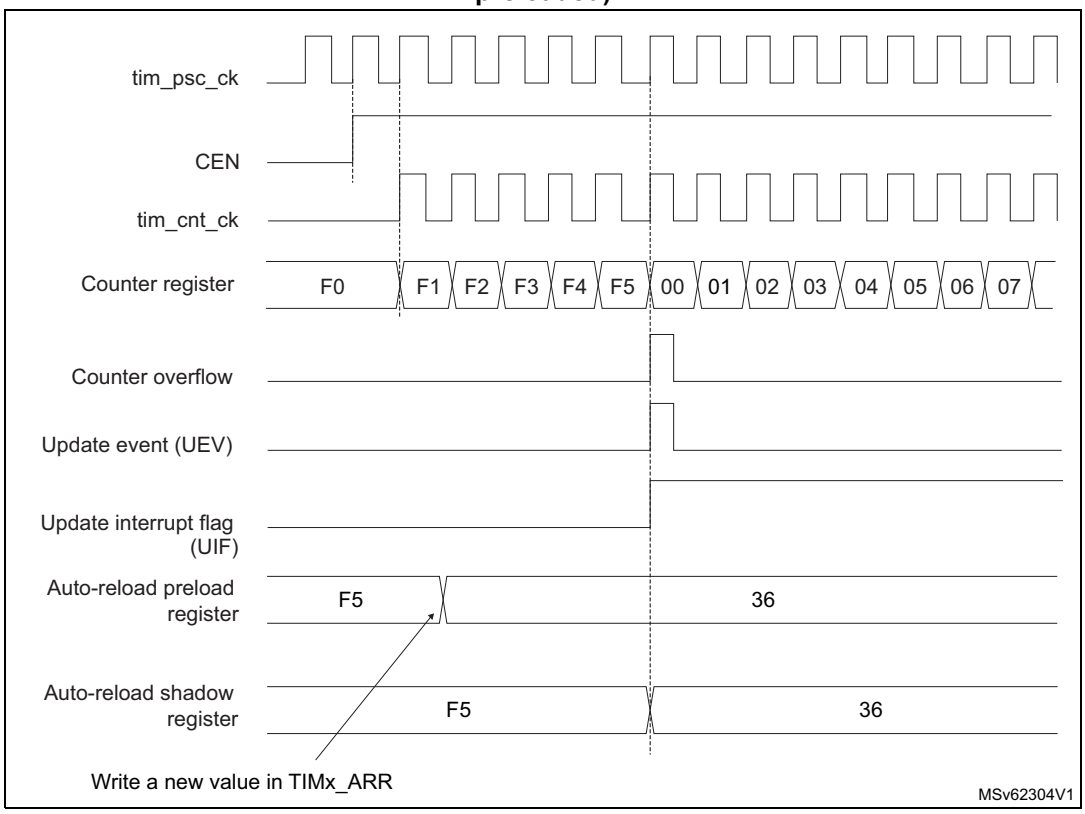


Figure 145. Counter timing diagram, Update event when ARPE = 1 (TIMx_ARR preloaded)



Down-counting mode

In down-counting mode, the counter counts from the autoreload value (content of the TIMx_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

An update event can be generated at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller)

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate does not change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 146. Counter timing diagram, internal clock divided by 1

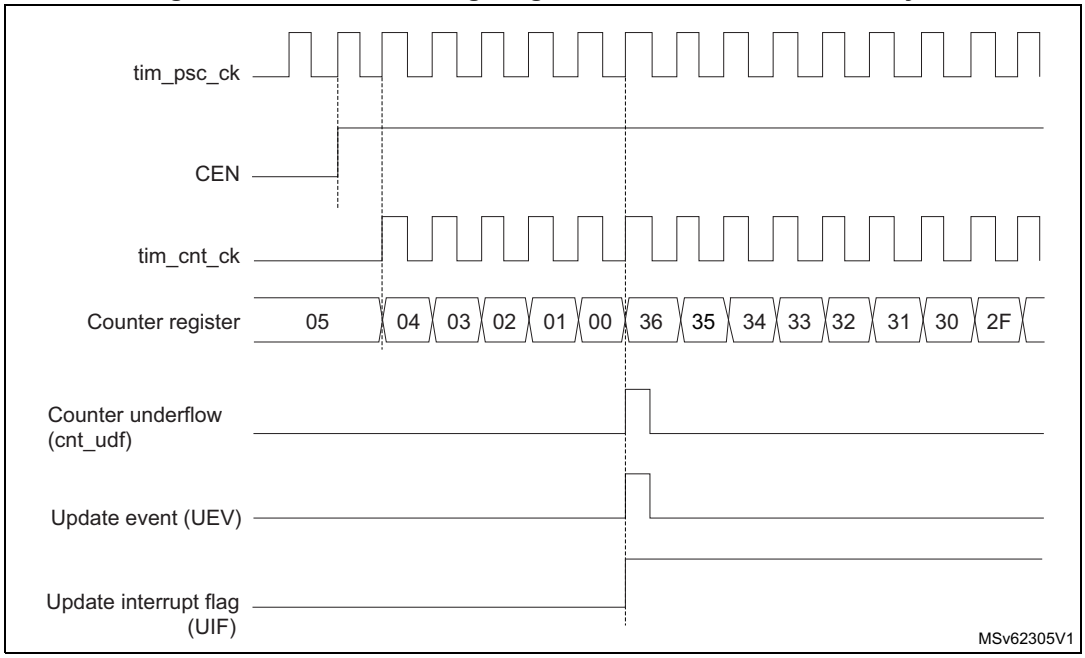


Figure 147. Counter timing diagram, internal clock divided by 2

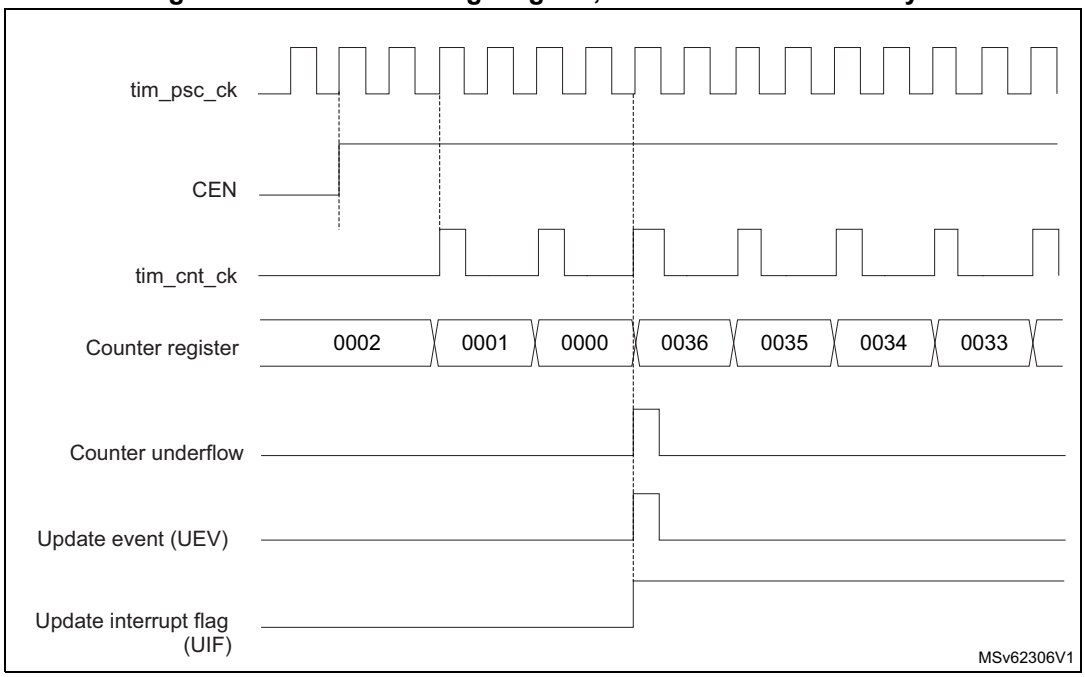


Figure 148. Counter timing diagram, internal clock divided by 4

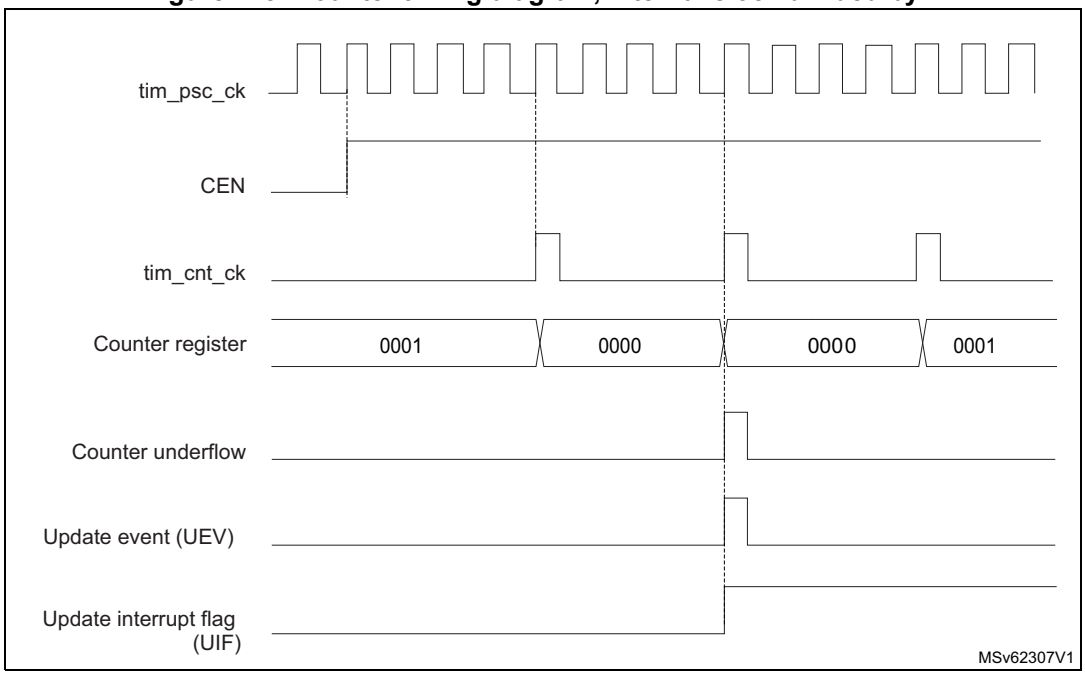


Figure 149. Counter timing diagram, internal clock divided by N

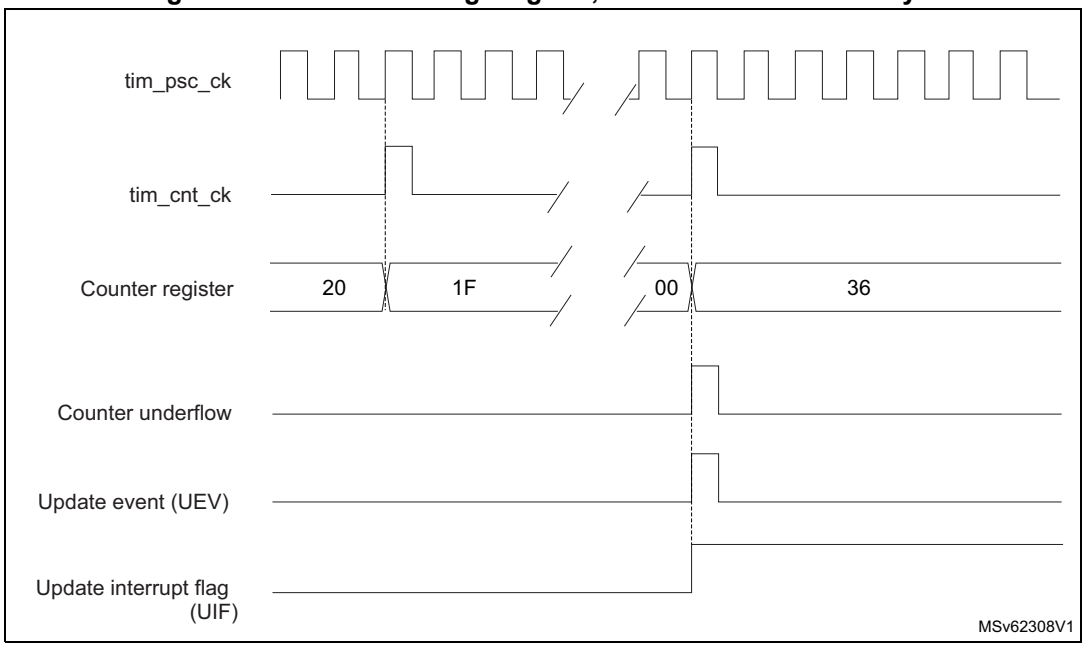
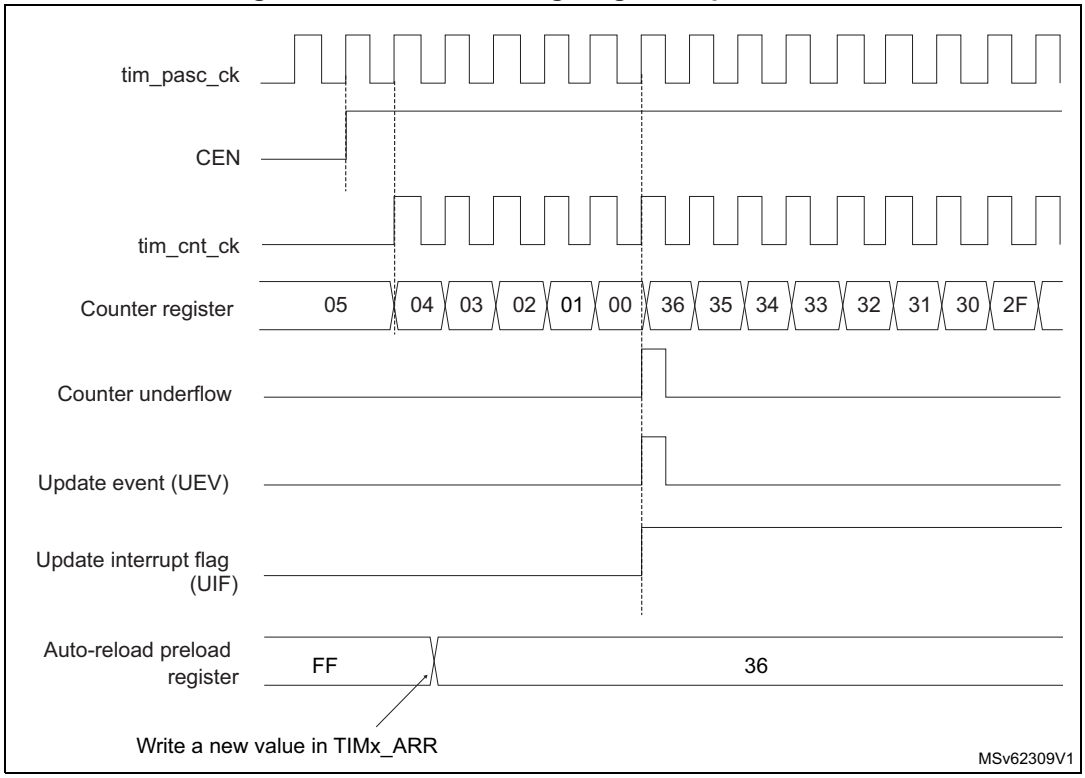


Figure 150. Counter timing diagram, Update event



Center-aligned mode (up/down-counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the

autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to 00. The output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = 01), the counter counts up (Center aligned mode 2, CMS = 10) the counter counts up and down (Center aligned mode 3, CMS = 11).

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

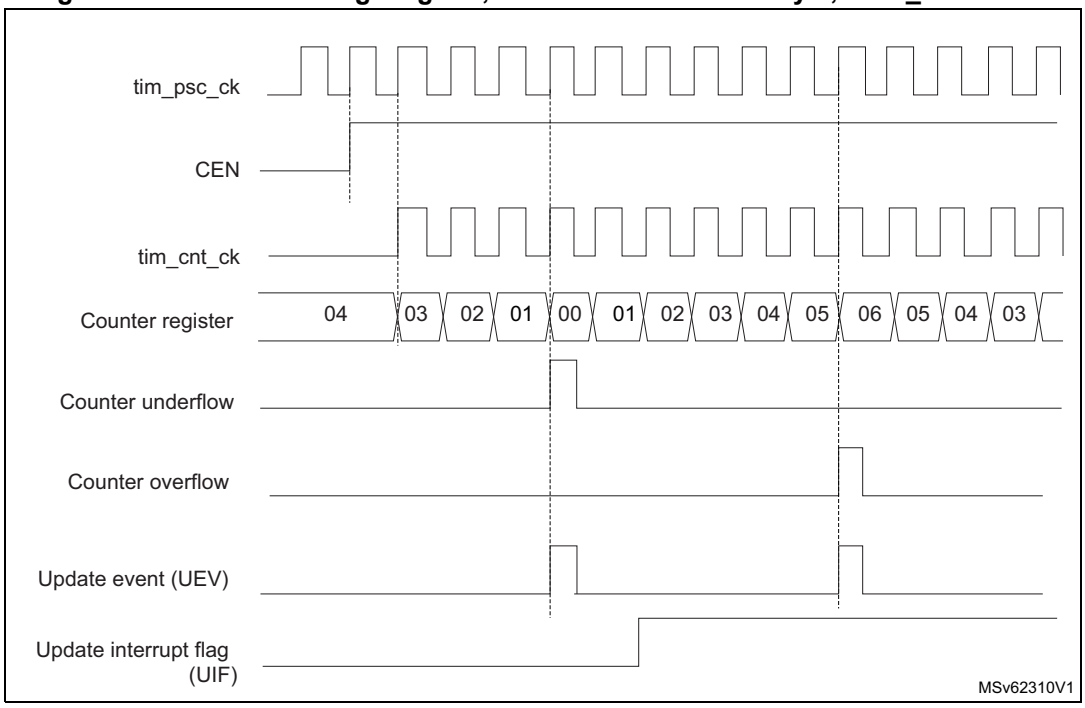
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 151. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 28.5.1: TIM2 control register 1 (TIM2_CR1)*).

Figure 152. Counter timing diagram, internal clock divided by 2

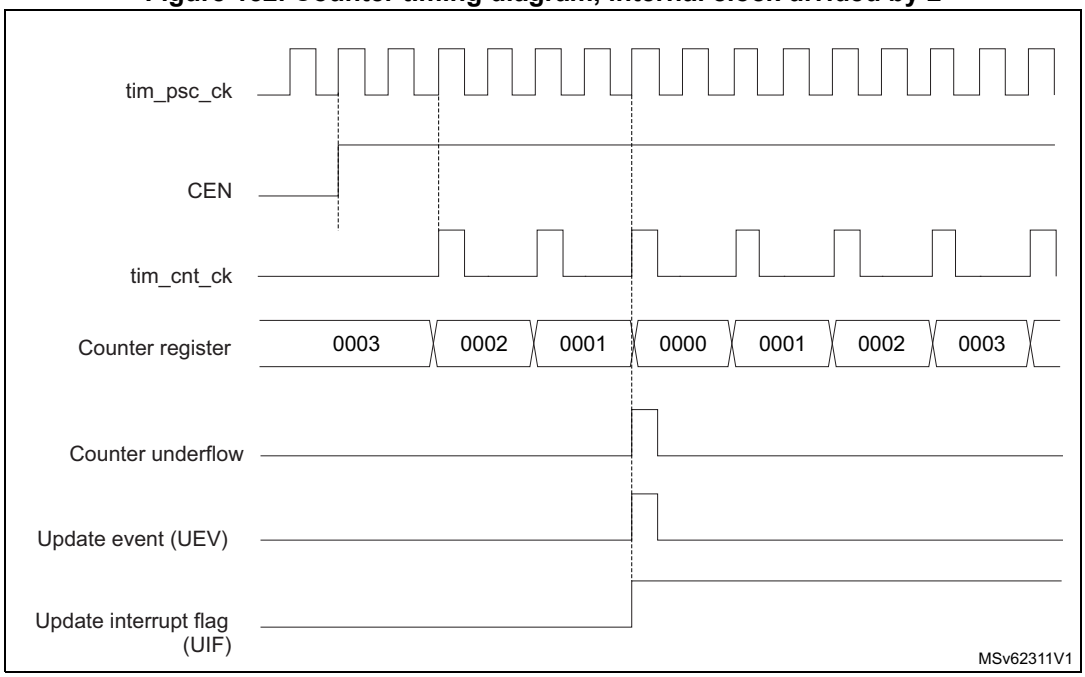
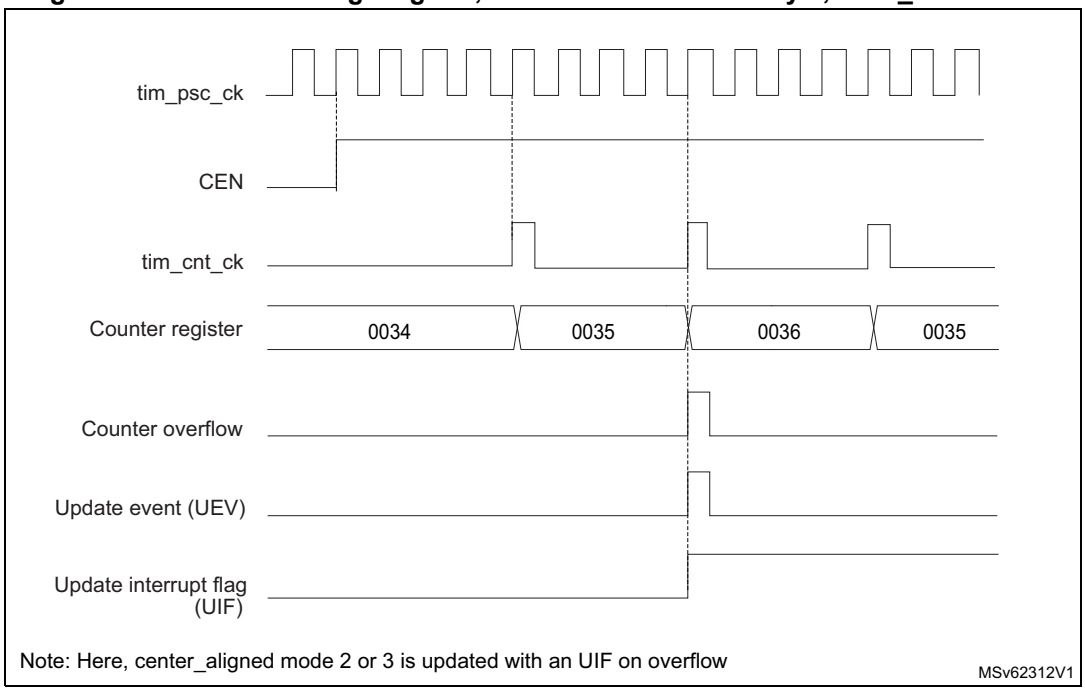


Figure 153. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36



1. Center-aligned mode 2 or 3 is used with a UIF on overflow.

Figure 154. Counter timing diagram, internal clock divided by N

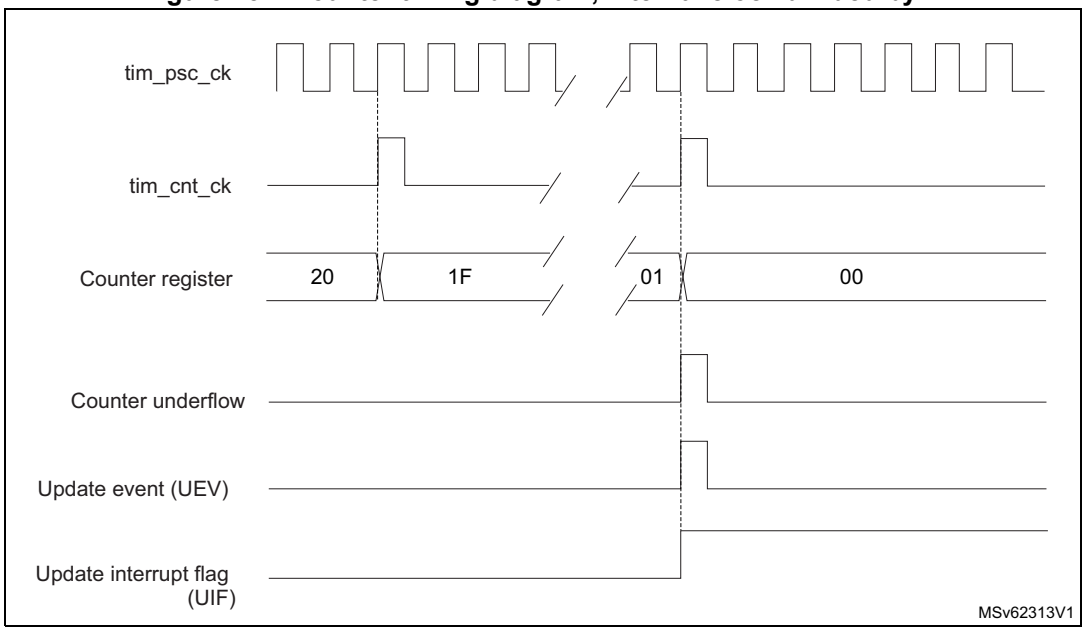


Figure 155. Counter timing diagram, Update event with ARPE = 1 (counter underflow)

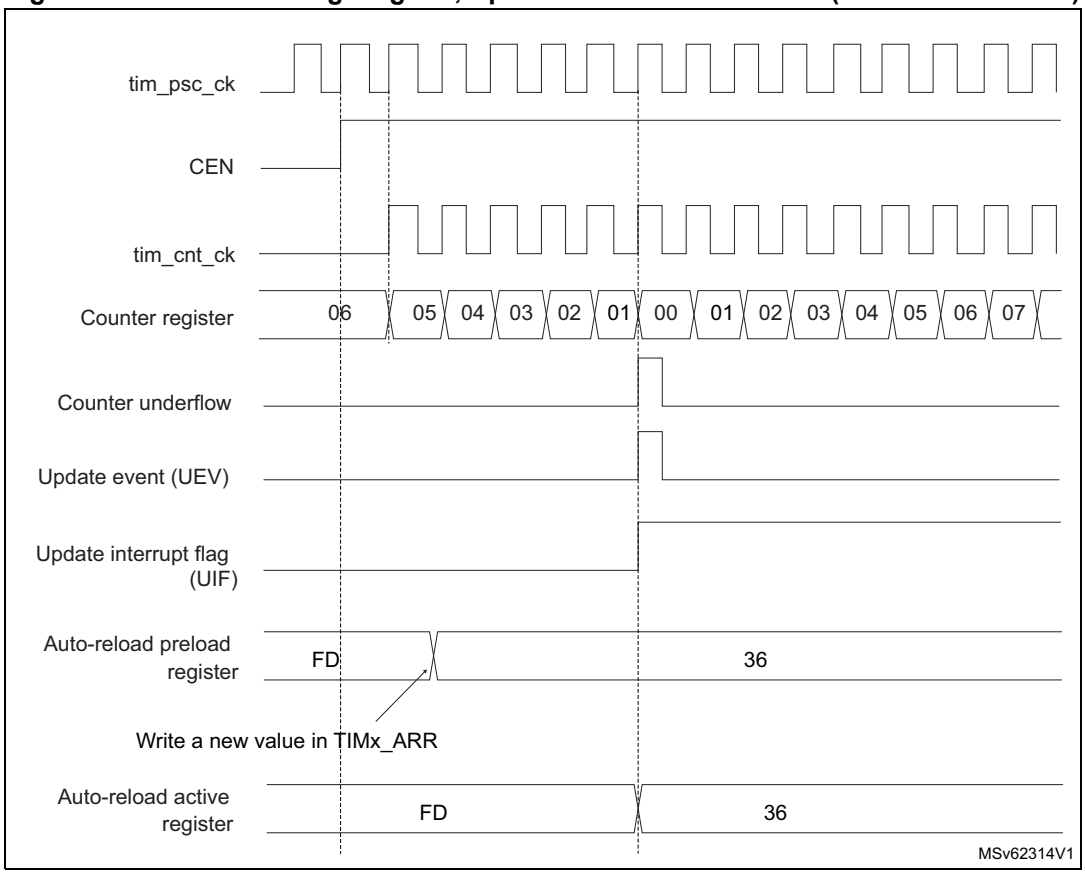
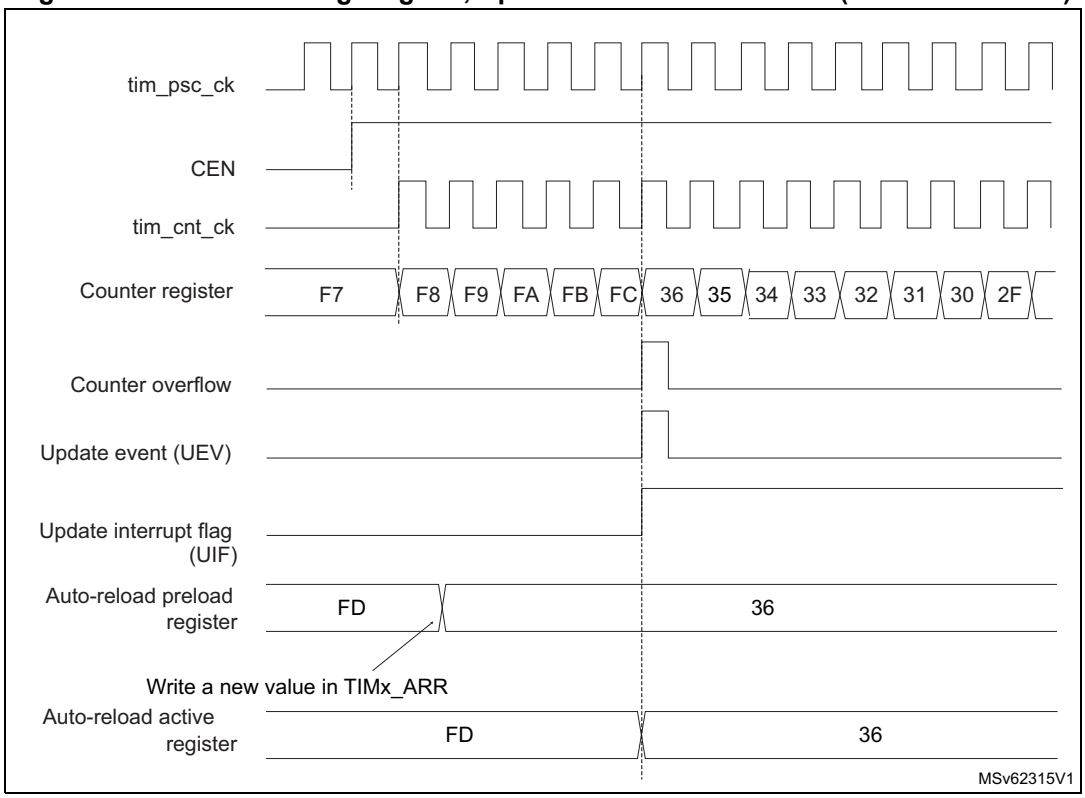


Figure 156. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



28.4.5 Clock selection

The counter clock can be provided by the following clock sources:

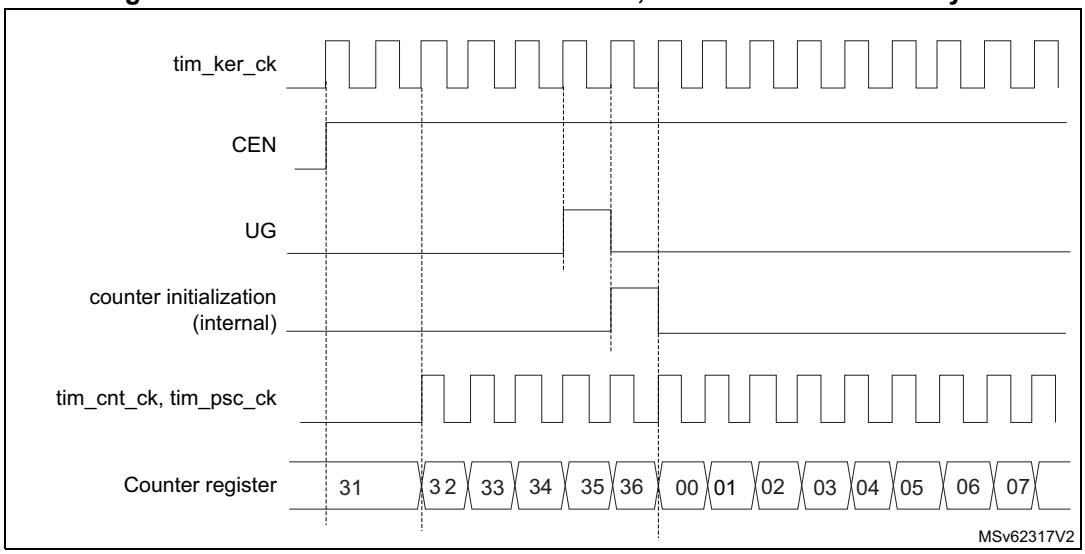
- Internal clock (tim_ker_ck).
- External clock mode1: external input pin (tim_ti1 or tim_ti2).
- External clock mode2: external trigger input (tim_etr_in).
- Internal trigger inputs (tim_itr): using one timer as prescaler for another timer, for example, timer 1 can be configured to act as a prescaler for timer 2. Refer to *Using one timer as prescaler for another timer* for more details.

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000 in the TIMx_SMCR register), then the CEN, DIR (in the TIMx_CR1 register), and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 157 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

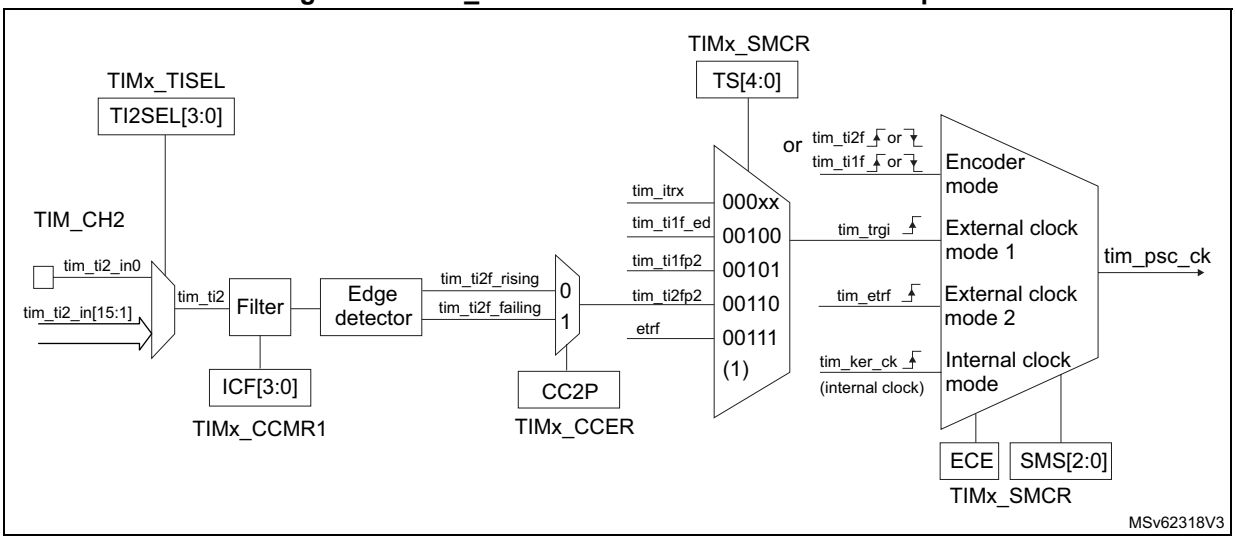
Figure 157. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 158. tim_ti2 external clock connection example



1. Codes ranging from 01000 to 11111: tim_itr[15:0].

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S= 01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).

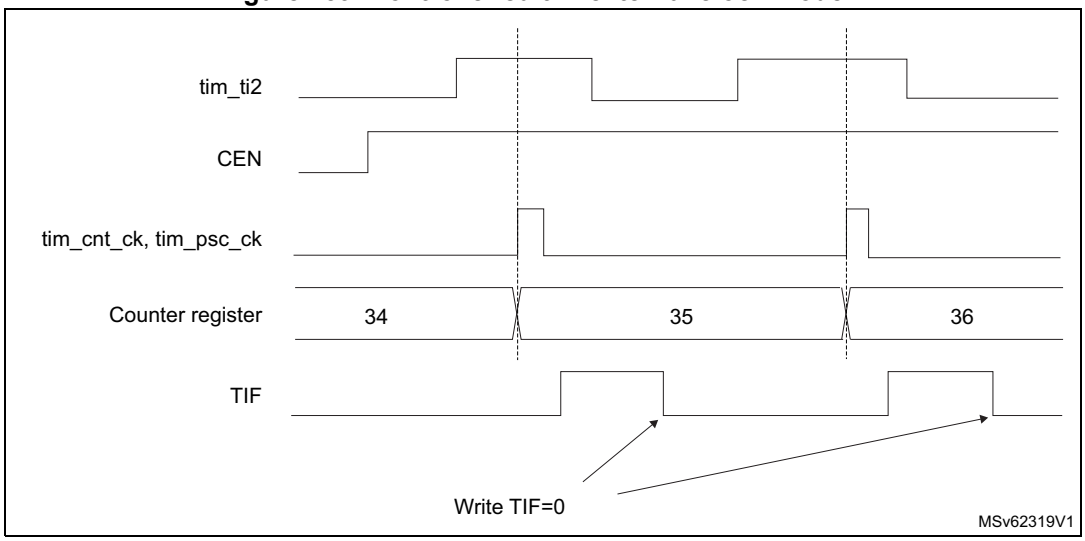
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

4. Select rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
6. Select tim_ti2 as the input source by writing TS = 00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 159. Control circuit in external clock mode 1



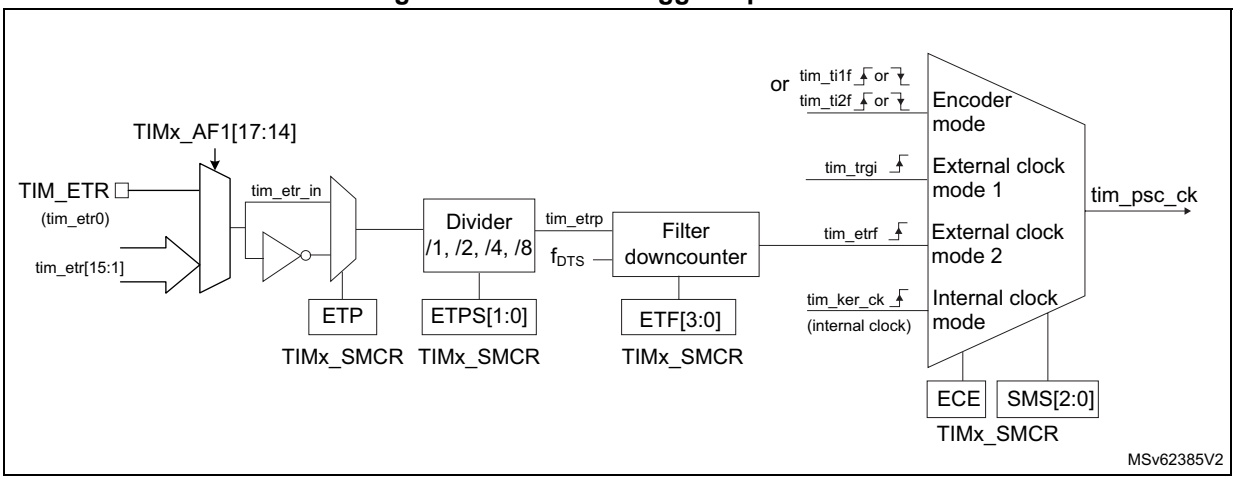
External clock source mode 2

This mode is selected by writing ECE = 1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input tim_etr_in.

Figure 160 gives an overview of the external trigger input block.

Figure 160. External trigger input block



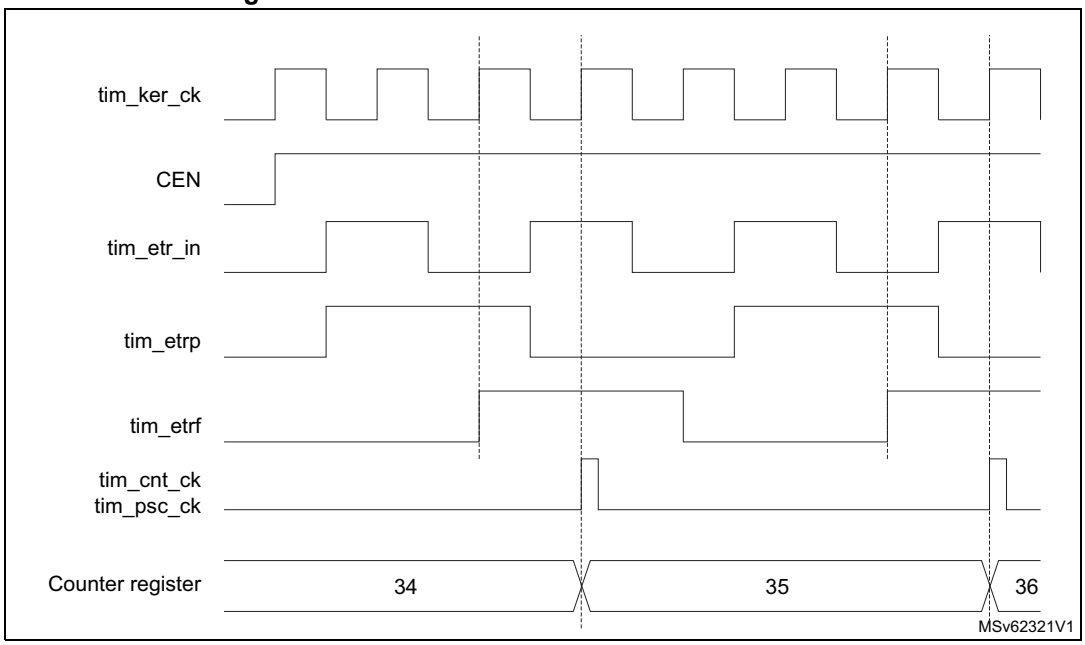
For example, to configure the upcounter to count each two rising edges on tim_etr_in, use the following procedure:

1. Select the proper tim_etr_in source (internal or external) with the ETRSEL[3:0] bits in the TIMx_AF1 register.
2. As no filter is needed in this example, write ETF[3:0] = 0000 in the TIMx_SMCR register.
3. Set the prescaler by writing ETPS[1:0] = 01 in the TIMx_SMCR register.
4. Select rising edge detection on the tim_etr_in by writing ETP = 0 in the TIMx_SMCR register.
5. Enable external clock mode 2 by writing ECE = 1 in the TIMx_SMCR register.
6. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter counts once each two tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency that can be correctly captured by the counter is at most ¼ of tim_ker_ck frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 161. Control circuit in external clock mode 2



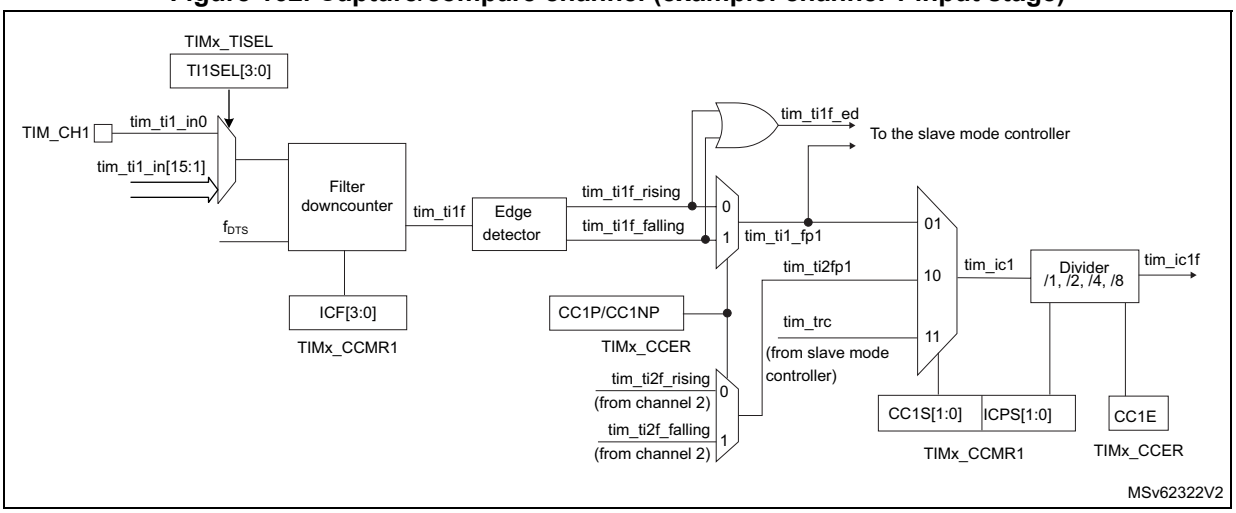
28.4.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one Capture/Compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 162. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 163. Capture/compare channel 1 main circuit

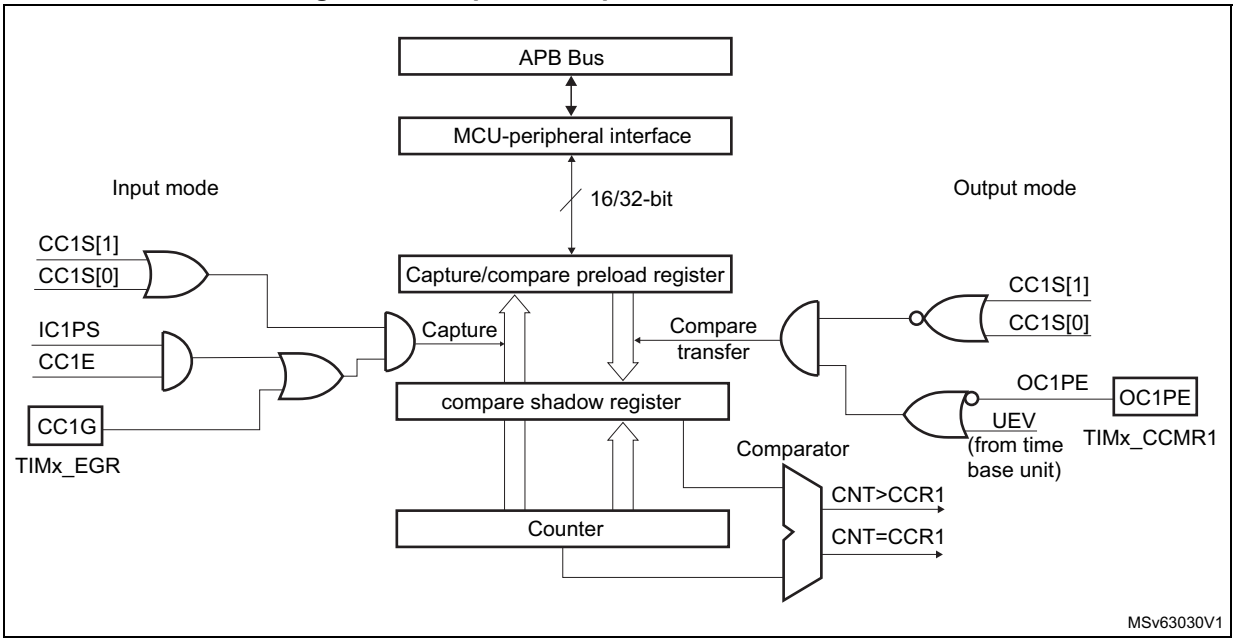
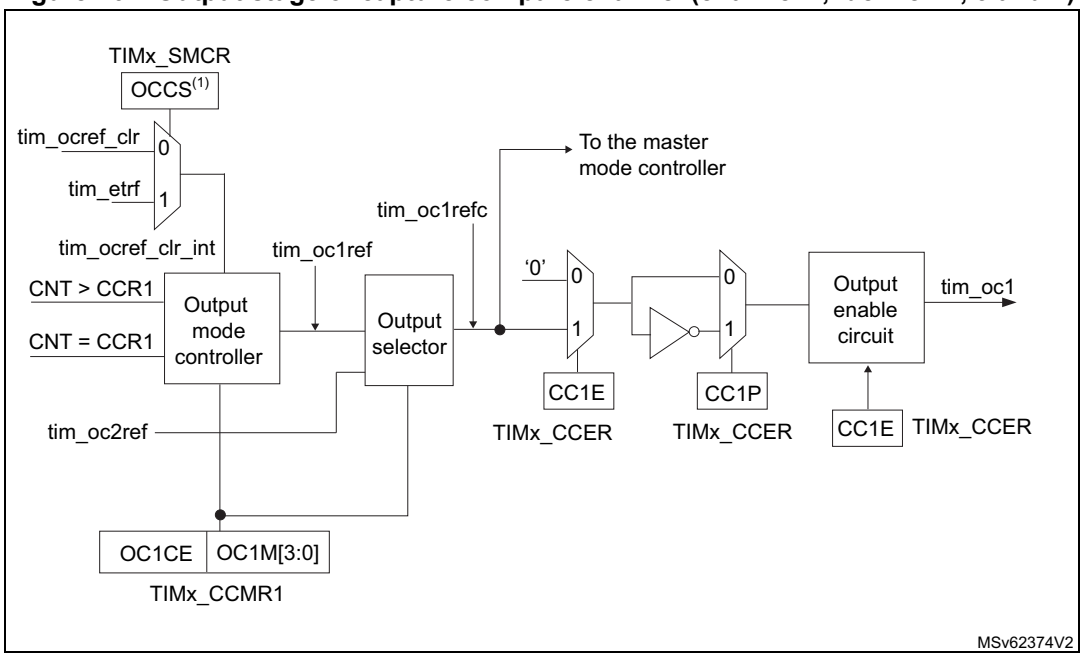


Figure 164. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4)



1. Available on some instances only. If not available, tim_etrf is directly connected to tim_ocref_clr_int.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

28.4.7 Input capture mode

In input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the needed input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by writing the CC1P and CC1NP bits to 000 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the capture is to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

28.4.8 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIMx_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges).

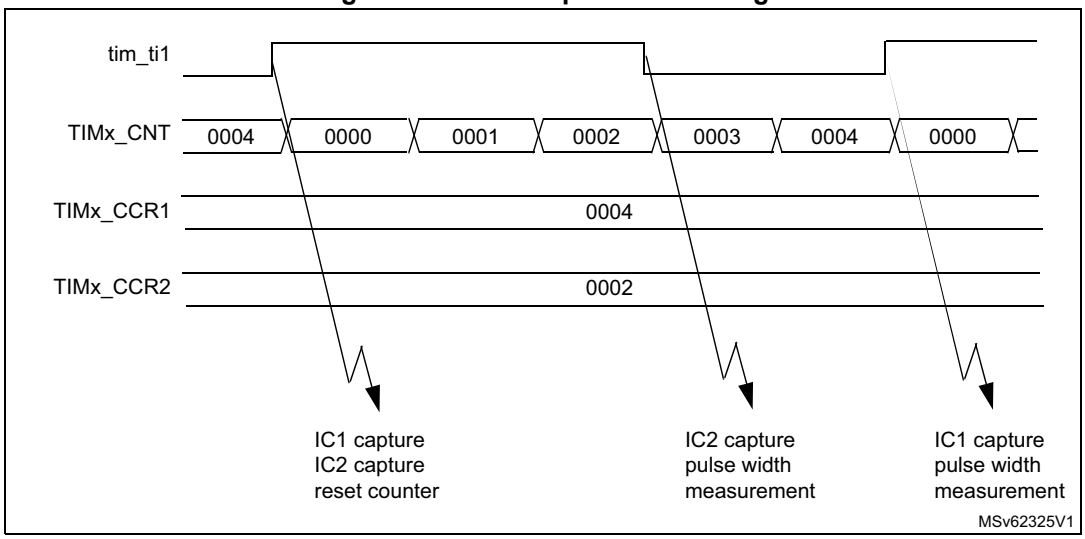
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tix input.
- These two ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulse width of a PWM signal applied on tim_ti1 can be measured using the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to 0 and the CC1NP bit to 0 (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P bit to 1 and the CC2NP bit to 0 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.

Figure 165. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

28.4.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, the user just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

tim_ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare mode section.

28.4.10 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM = 000), be set

active (OCxM = 001), be set inactive (OCxM = 010) or can toggle (OCxM = 011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

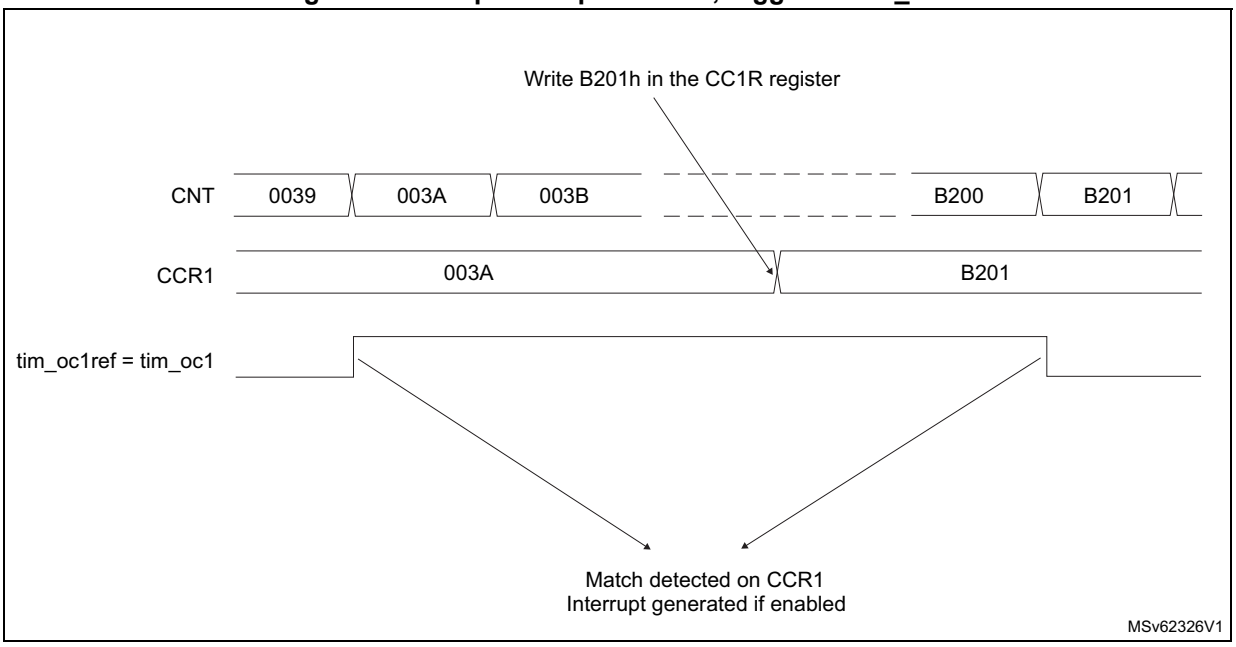
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example:
   a) Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx.
   b) Write OCxPE = 0 to disable preload register.
   c) Write CCxP = 0 to select active high polarity.
   d) Write CCxE = 1 to enable the output.
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 166*.

Figure 166. Output compare mode, toggle on tim_oc1



28.4.11 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 110 (PWM mode 1) or 111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in up-counting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter). The tim_ocref_clr can be cleared by an external event through the tim_etr_in or the tim_oceref_clr signals. In this case the tim_ocref_clr signal is asserted only:

- After a compare match event.
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM = 000) to one of the PWM modes (OCxM = 110 or 111). This forces the PWM by software while the timer is running.

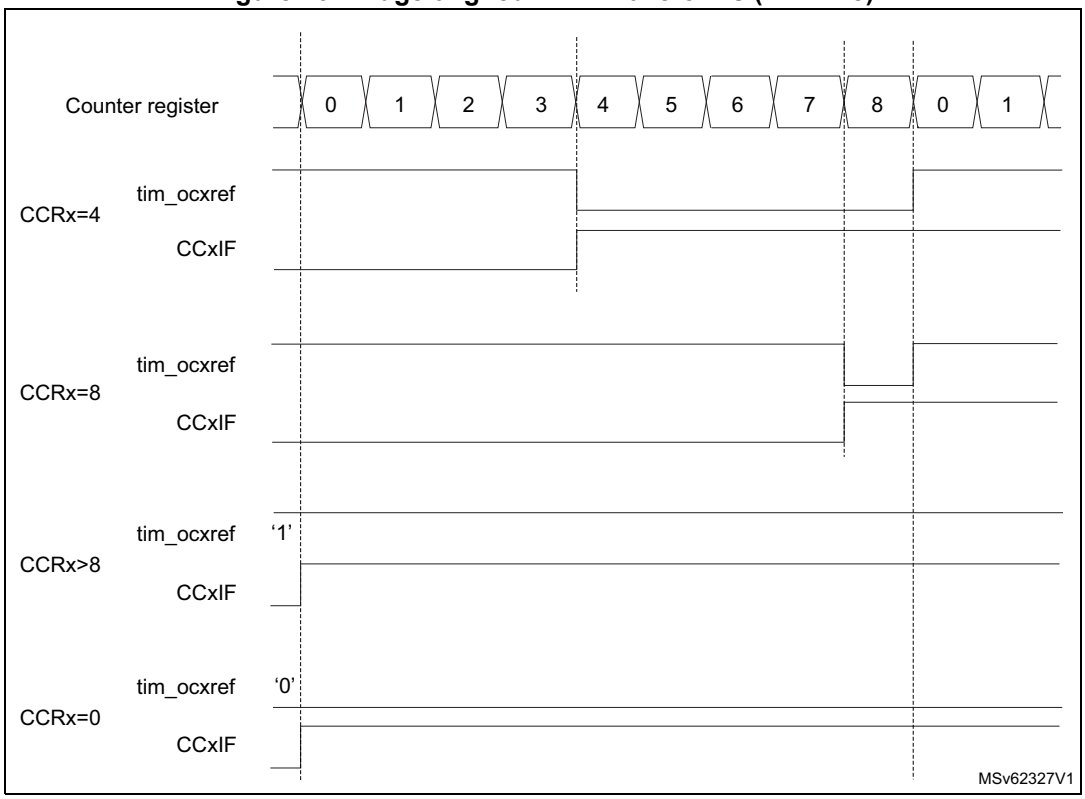
The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

PWM edge-aligned mode

- Up-counting configuration
- Up-counting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Up-counting mode*.

  In the following example, we consider PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT <TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. *Figure 167* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 167. Edge-aligned PWM waveforms (ARR = 8)



Down-counting configuration

- Down-counting is active when DIR bit in TIMx_CR1 register is high. Refer to *Down-counting mode*.

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIMx_CNT>TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the autoreload value in TIMx_ARR, then tim_ocxref is held at 100%. PWM is not possible in this mode.

PWM center-aligned mode

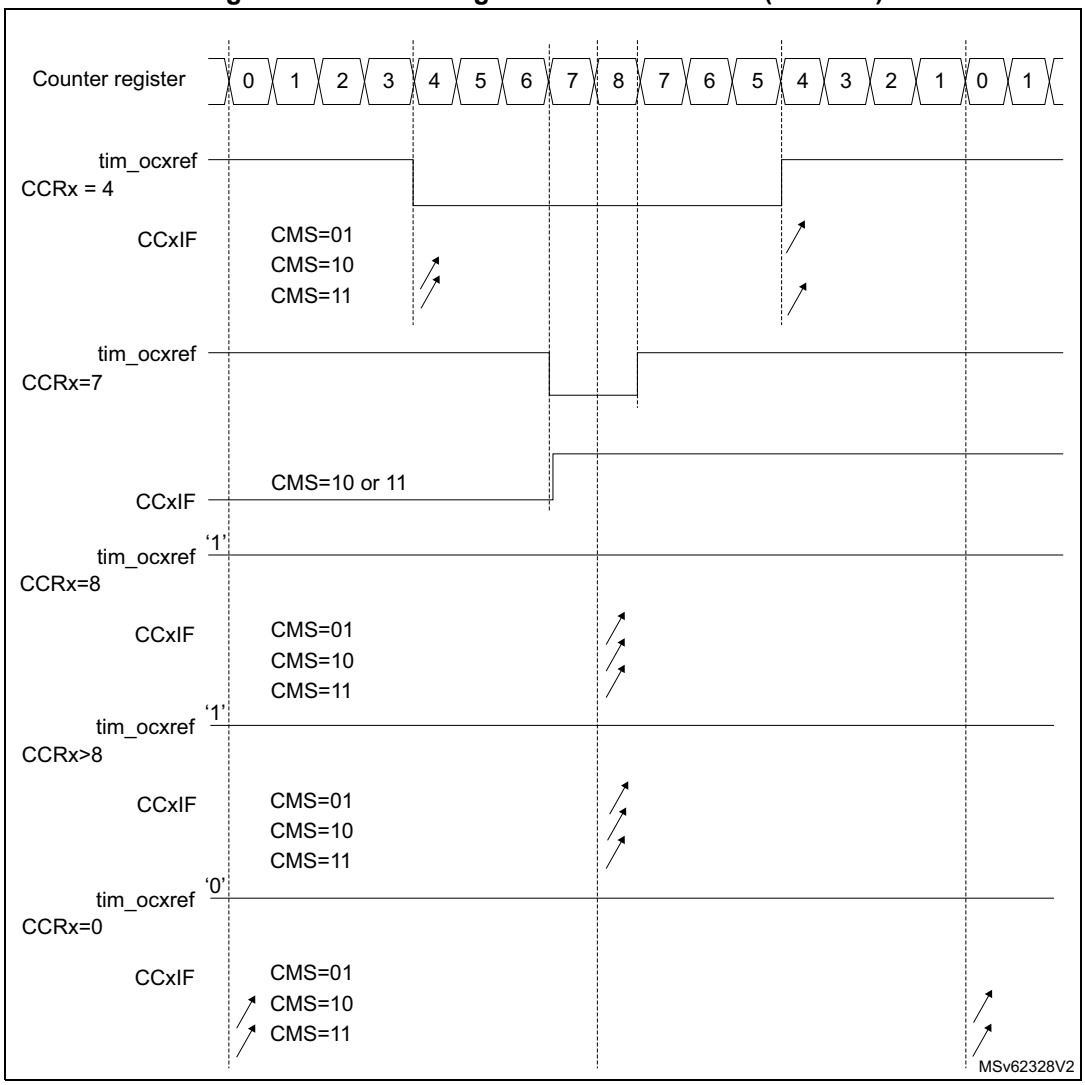
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit

(DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down-counting)*.

Figure 168 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR = 8.
- PWM mode is the PWM mode 1.
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIMx_CR1 register.

Figure 168. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

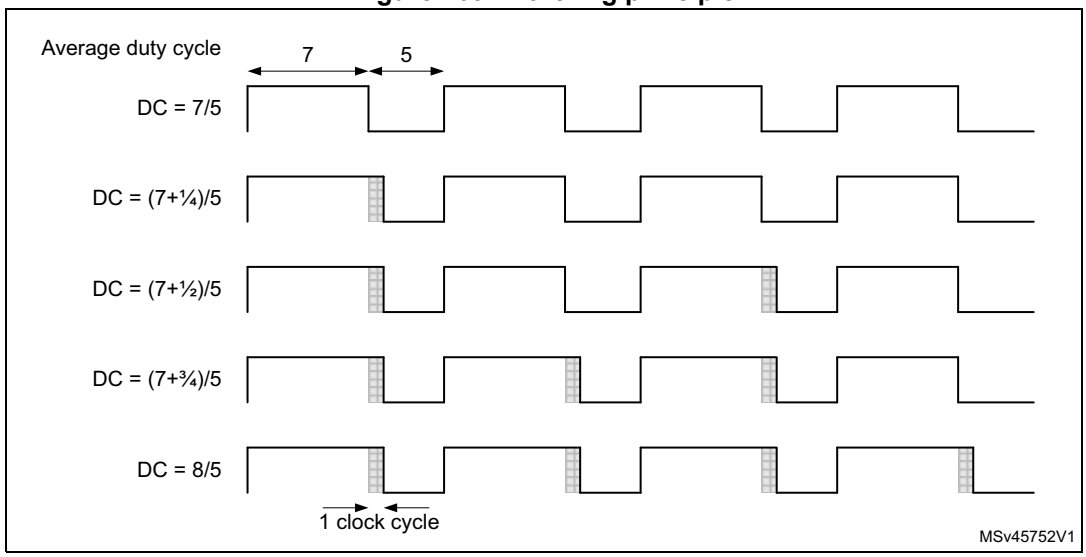
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no update event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 169* presents the dithering principle applied to four consecutive PWM cycles.

Figure 169. Dithering principle



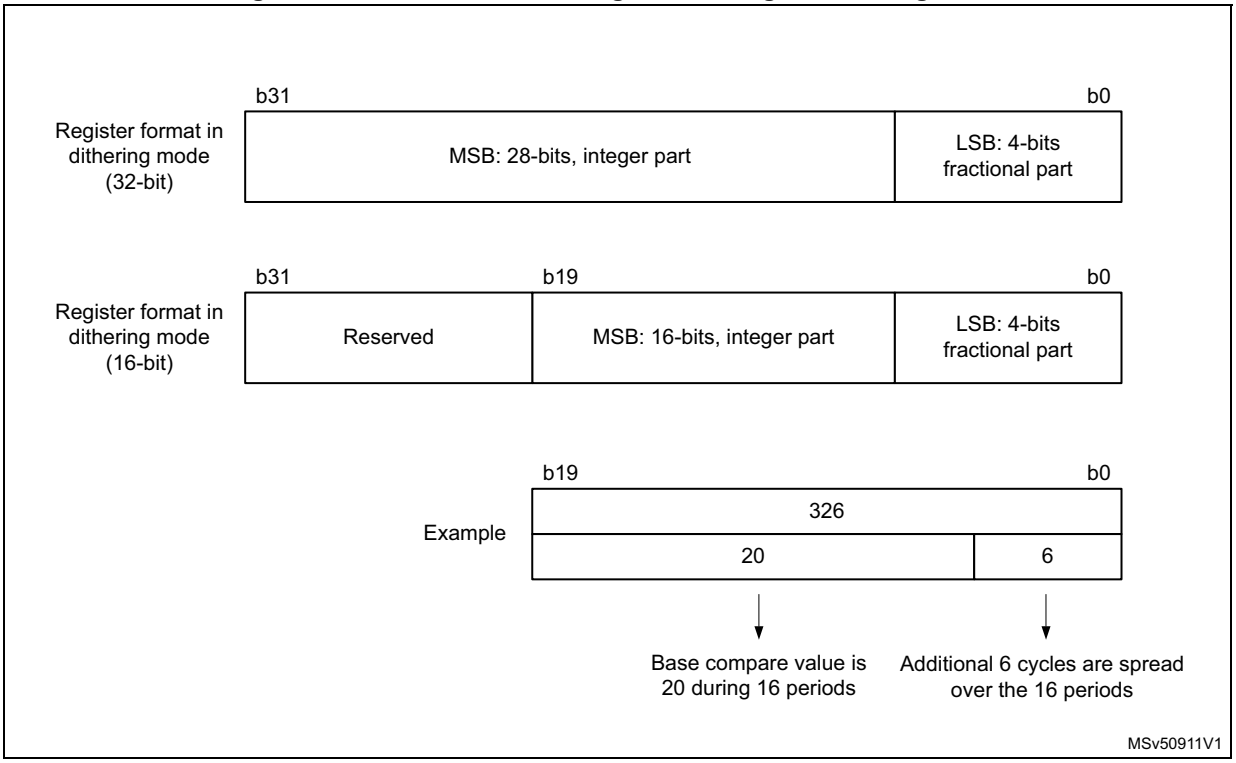
When the dithering mode is enabled, the register coding is changed as following (see *Figure 170* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted by four places and are coding for the base value. In 16-bit mode, the 16-bit format is maintained.

Note: The following sequence must be followed when resetting the DITHEN bit:

1. *CEN and ARPE bits must be reset.*
2. *The DITHEN bit must be reset.*
3. *The CCIF flags must be cleared.*
4. *The CEN bit can be set (eventually with ARPE = 1).*

Figure 170. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode (16-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

$$\text{Dithering mode (32-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{268435454 + \frac{15}{16}}$$

Note: For 16-bit timers, the maximum TIMx_ARR and TIMx_CCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part). For 32-bit timers, the maximum TIMx_ARR and TIMx_CCRy values are limited to

0xFFFFFFEF in dithering mode (corresponds to 264435454 for the integer part and 15 for the dithered part).

As shown on *Figure 171* and *Figure 172*, the dithering mode is used to increase the PWM resolution.

Figure 171. PWM resolution vs frequency (16-bit mode)

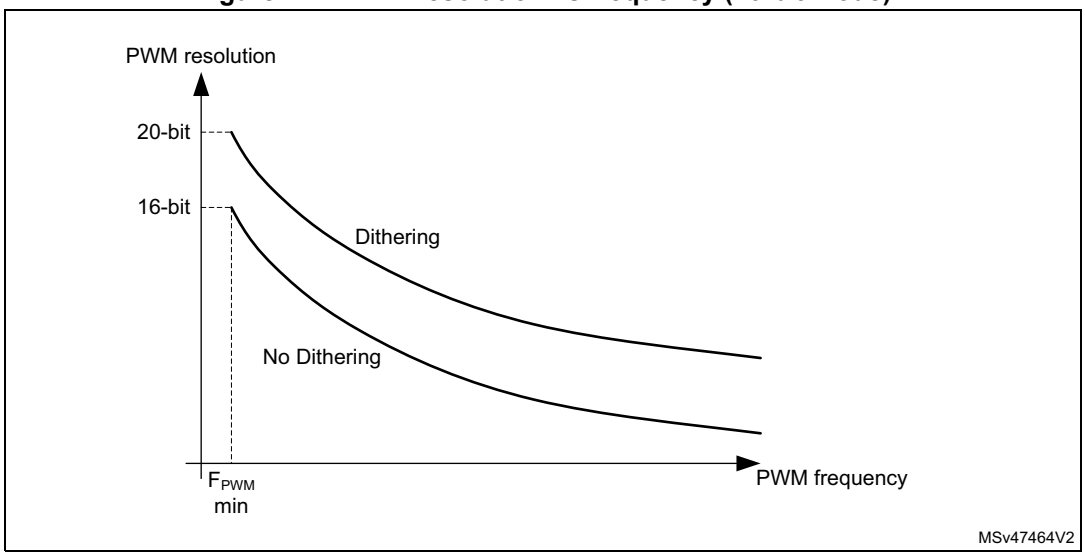
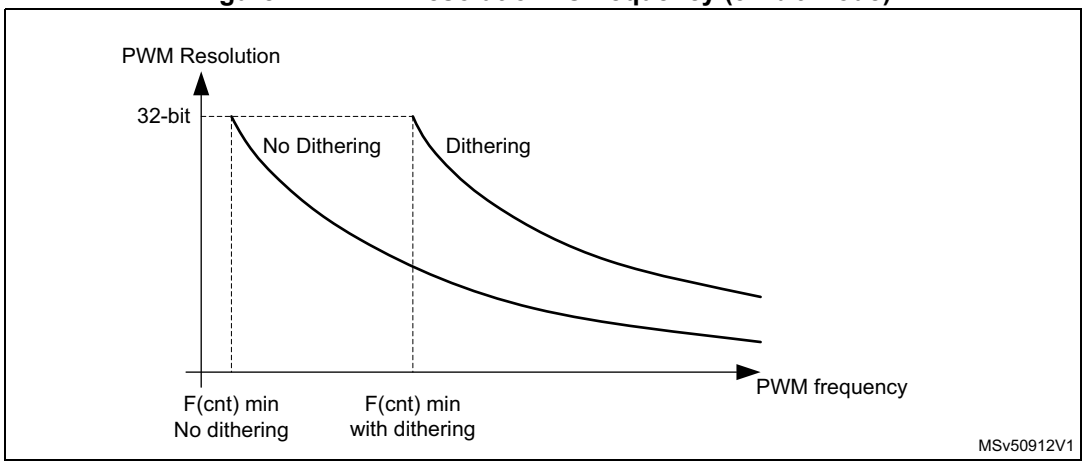
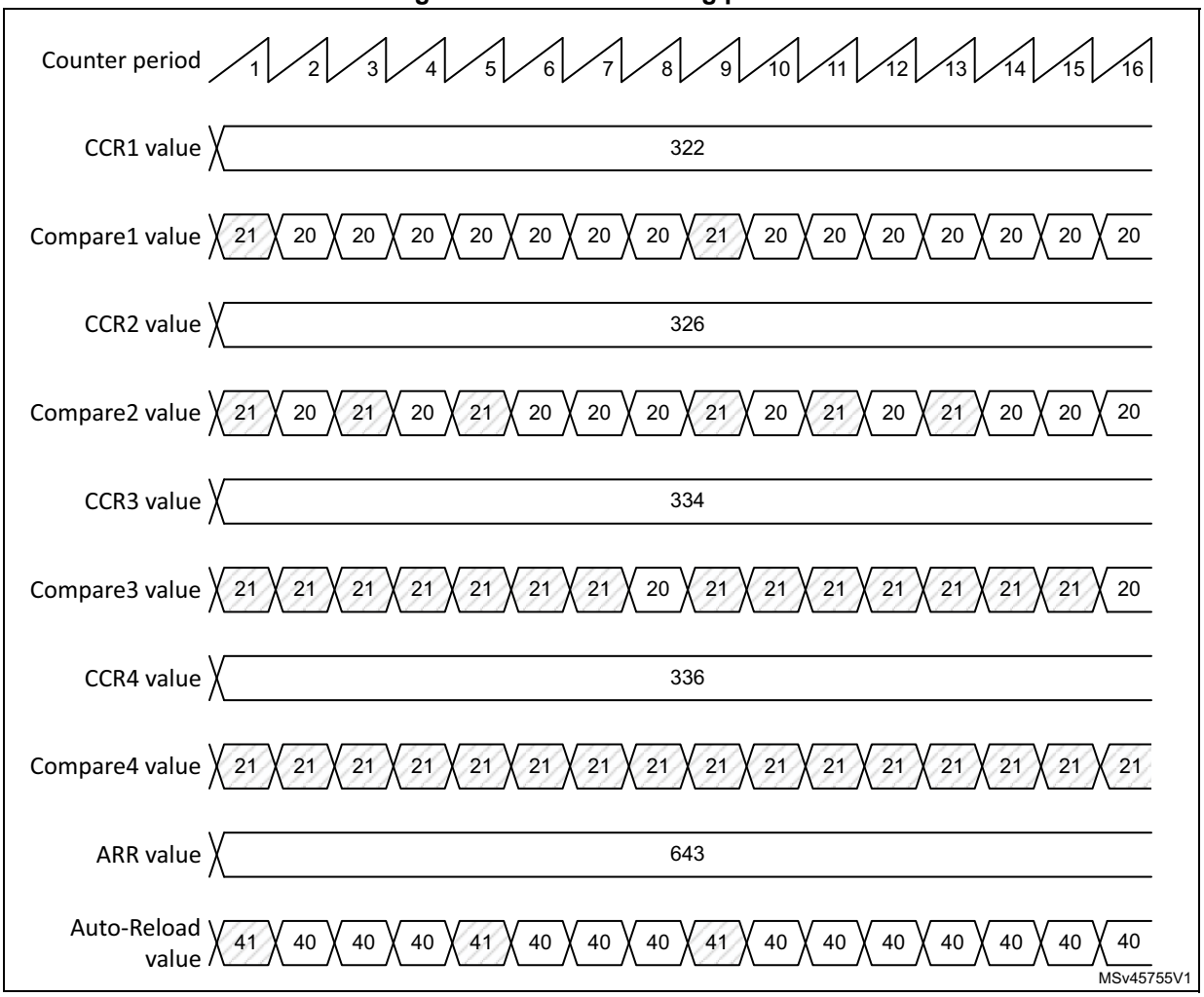


Figure 172. PWM resolution vs frequency (32-bit mode)



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 173*.

Figure 173. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 258*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 258. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIMx_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down-counting phases as shown in *Figure 174*.

Figure 174. Dithering effect on duty cycle in center-aligned PWM mode

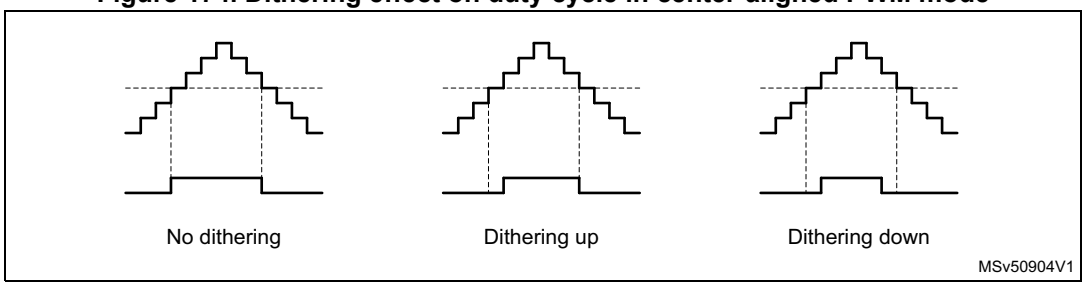


Table 259 shows how the dithering pattern is added in center-aligned PWM mode.

Table 259. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

28.4.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx registers. One register controls the PWM during up-counting, the second during down-counting, so that PWM is adjusted every half PWM cycle:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2.
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4.

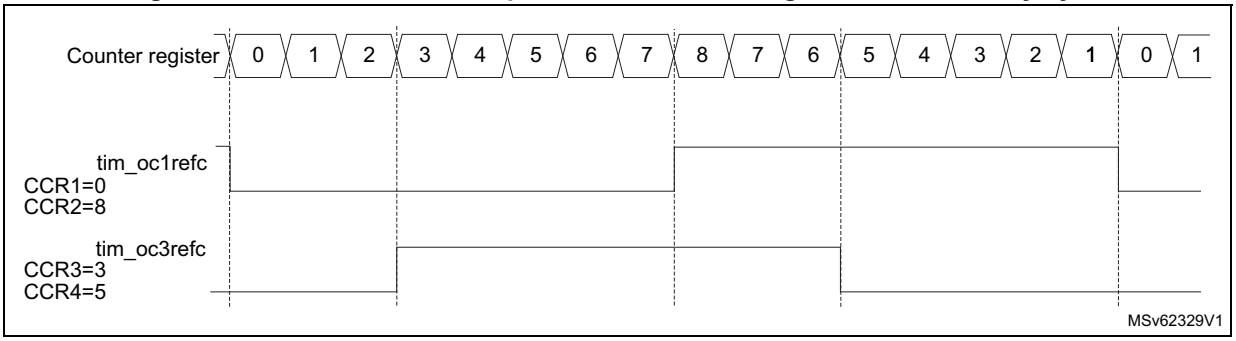
Asymmetric PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1110 (Asymmetric PWM mode 1) or 1111 (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the tim_oc2ref signal on channel 2, or an tim_oc2refc signal resulting from asymmetric PWM mode 2.

Figure 175 shows an example of signals that can be generated using asymmetric PWM mode (channels 1 to 4 are configured in asymmetric PWM mode 2).

Figure 175. Generation of two phase-shifted PWM signals with 50% duty cycle



28.4.13 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

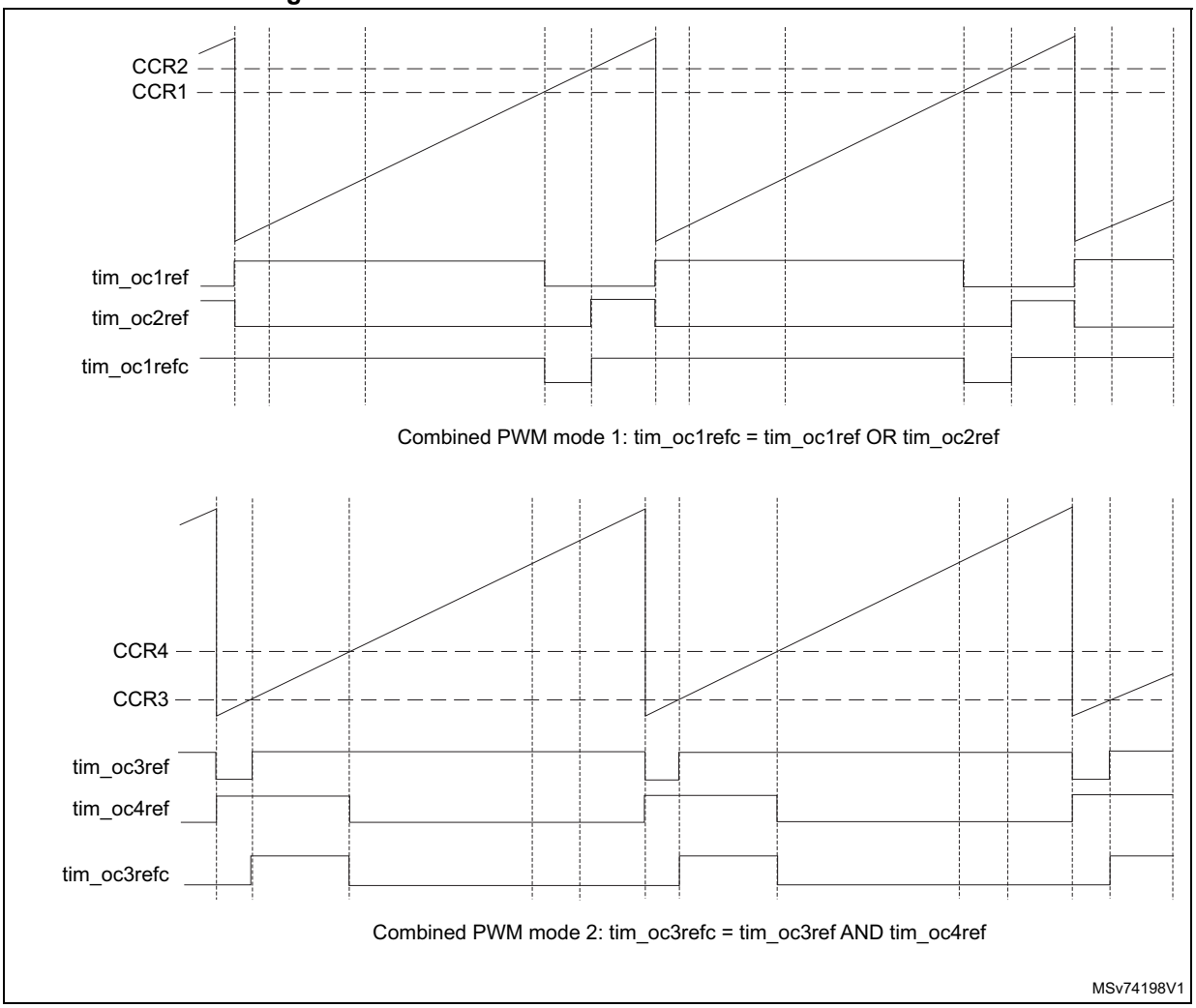
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 176 shows an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 1.
- Channel 2 is configured in PWM mode 2.
- Channel 3 is configured in Combined PWM mode 2.
- Channel 4 is configured in PWM mode 1.

Figure 176. Combined PWM mode on channels 1 and 3



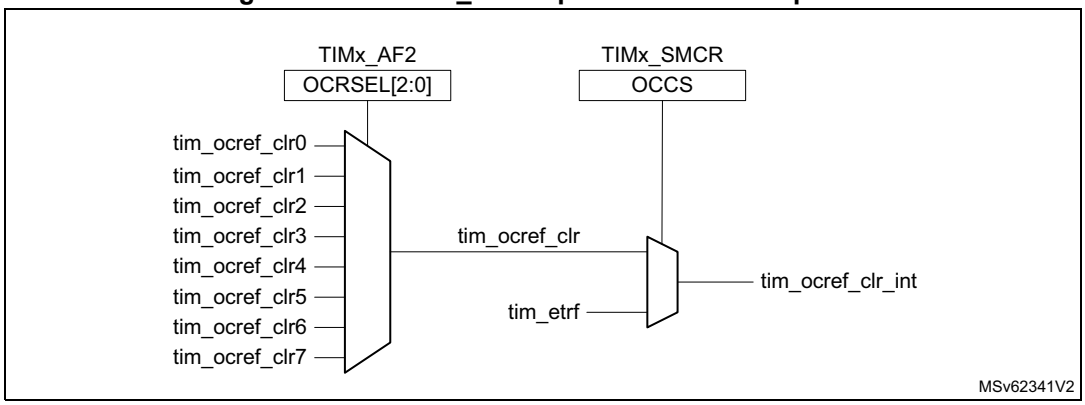
28.4.14 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The tim_ocref_clr_int source depends on the OCREF clear selection feature implementation, refer to *Section 28.3: TIM2 implementation*.

If the OCREF clear selection feature is implemented, the tim_ocref_clr_int can be selected between the tim_ocref_clr input and the tim_etrf input (tim_etr_in after the filter) by configuring the OCCS bit in the TIMx_SMCR register. The tim_ocref_clr input can be selected among several tim_ocref_clr[7:0] inputs, using the OCRSEL[2:0] bitfield in the TIMx_AF2 register, as shown in *Figure 177*.

Figure 177. OCREF_CLR input selection multiplexer



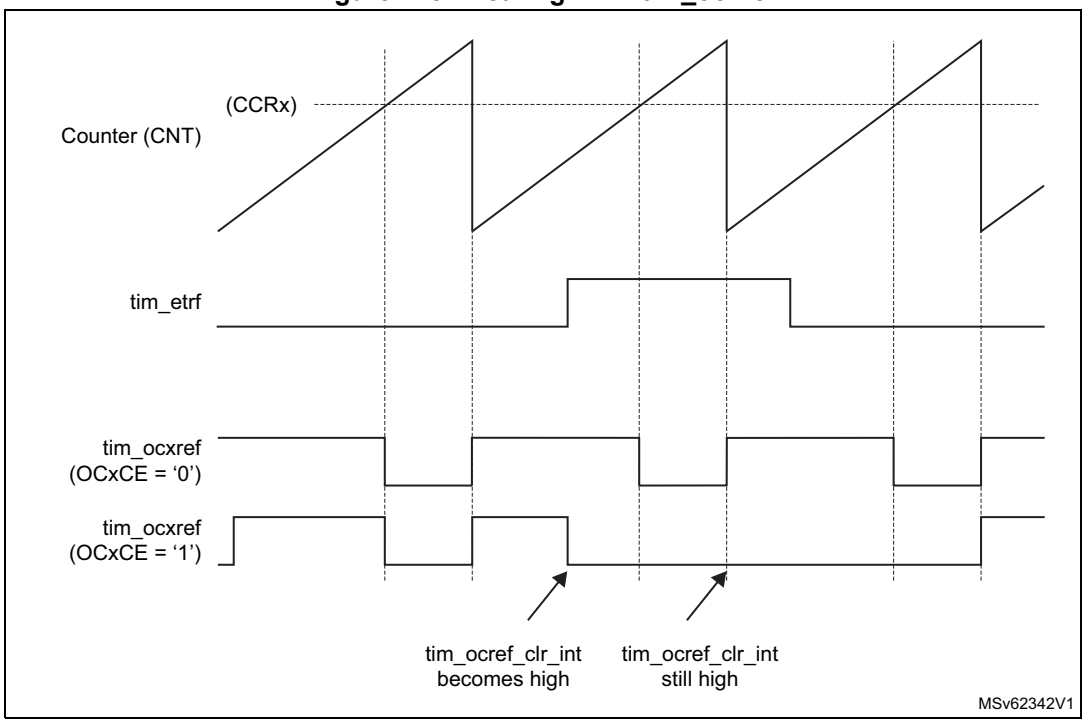
If the OCREF clear selection feature is not implemented, the tim_ocref_clr_int input is directly connected to the tim_etrf input.

For example, the tim_ocref_clr_int signal can be connected to the output of a comparator to be used for current handling. In this case, tim_etr_in must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIMx_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

Figure 178 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

Figure 178. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), tim_ocxref is enabled again at the next counter overflow.

28.4.15 One-pulse mode

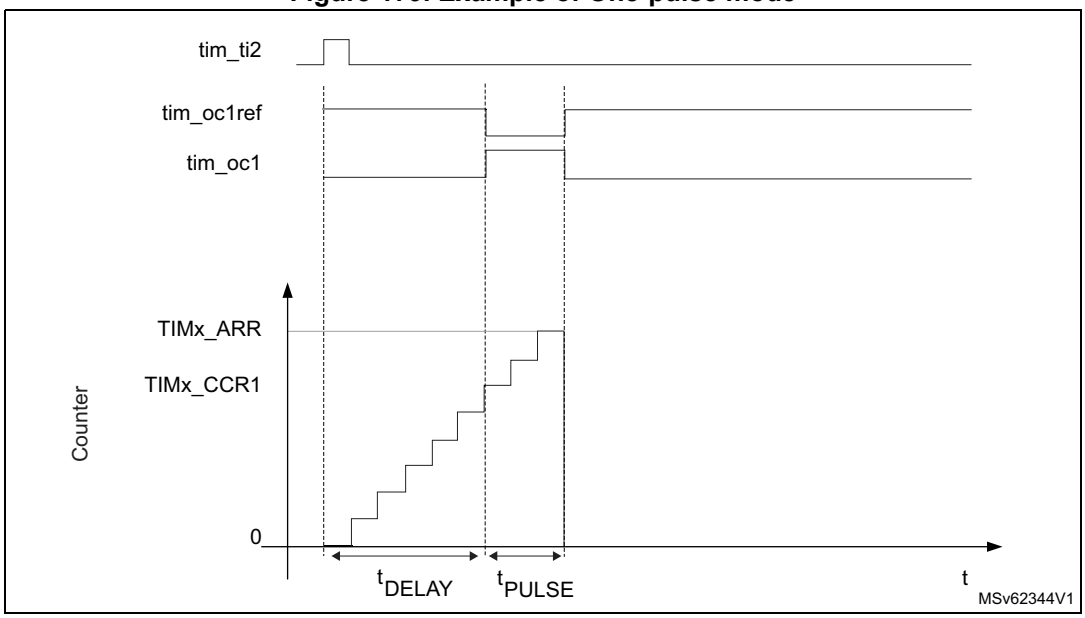
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

CNT<CCRx ≤ ARR (in particular, 0<CCRx).

Figure 179. Example of One-pulse mode



For example if the user wants to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 on tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Suppose the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

In this example, the DIR and CMS bits in the TIMx_CR1 register must be low.

Since only one pulse (Single mode) is needed, a one must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0). When OPM bit in the TIMx_CR1 register is set to 0, so the Repetitive mode is selected.

Particular case: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

28.4.16 Retriggerable one-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one-pulse mode described in *Section 28.4.15*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for Retriggerable OPM mode 1 or 2.

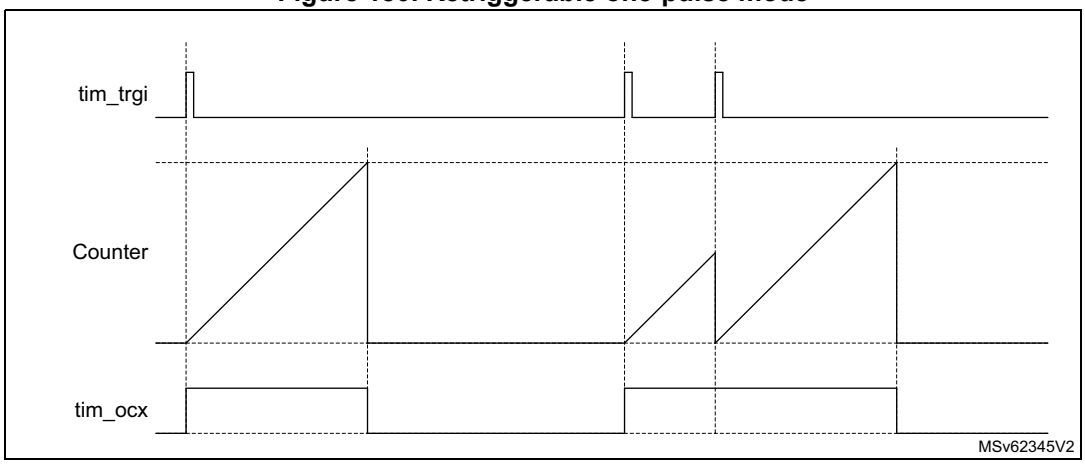
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode CCRx must be above or equal to ARR.

Note: In Retriggerable one-pulse mode, the CCxIF flag is not significant.

The OCxM[3:0] and SMS[3:0] bitfields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 180. Retriggerable one-pulse mode



28.4.17 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulse width generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on *Figure 181*.

Figure 181. Pulse generator circuitry

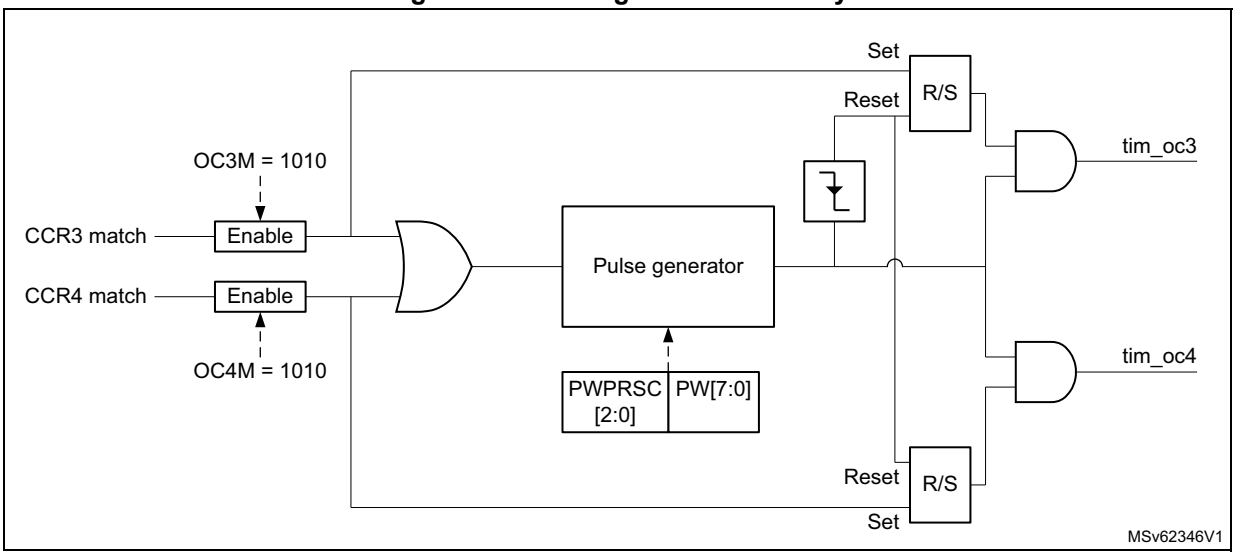
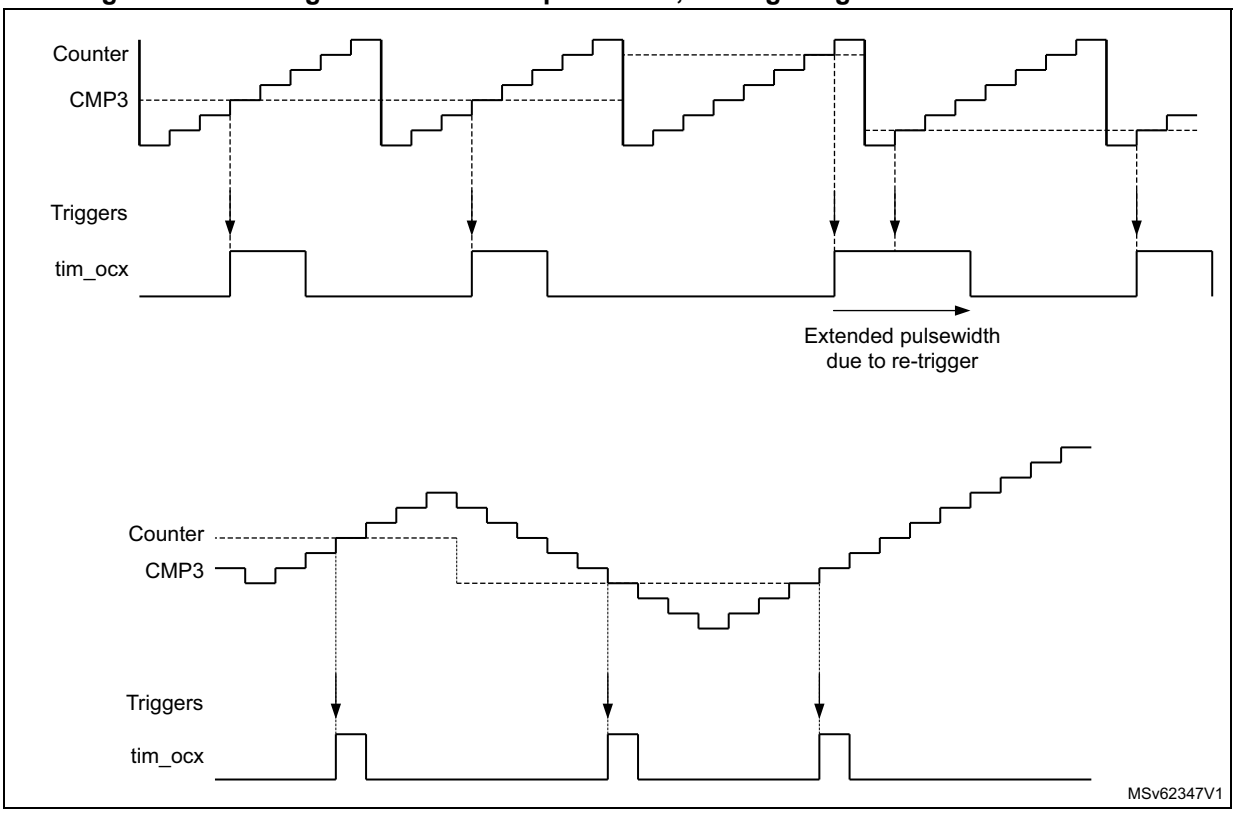


Figure 182 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 182. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bitfields in TIMx_CCMR2 register.

The pulse width is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

$t_{PW} = PW[7:0] \times t_{PWG}$

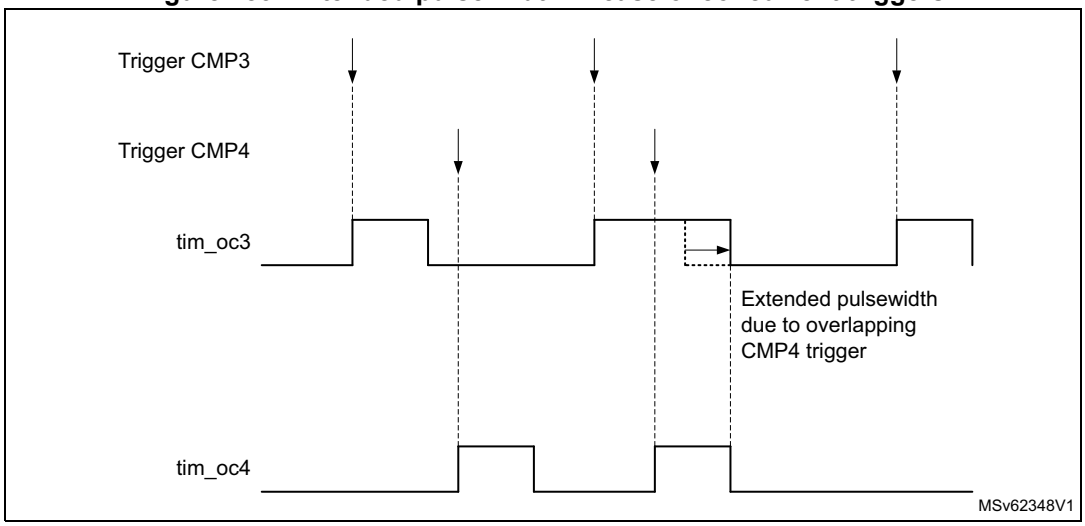
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$.

gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the first arriving trigger is extended (because of the retrigger), while the pulse width of the last arriving trigger is correct (as shown on Figure 183).

Figure 183. Extended pulse width in case of concurrent triggers



28.4.18 Encoder interface mode

Quadrature encoder

To select Encoder interface mode write SMS = 0001 in the TIMx_SMCR register if the counter is counting on tim_ti1 edges only, SMS = 0010 if it is counting on tim_ti2 edges only and SMS = 0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well.

The two inputs tim_ti1 and tim_ti2 are used to interface to an incremental encoder. Refer to *Table 260*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1 = tim_ti1 if not filtered and not inverted, tim_ti2fp2 = tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or

tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction corresponds to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 260. Counting direction versus encoder signals(CC1P = CC2P = 0)

Active edge	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	1110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	1111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	0001	High	Down	Up	No count	No count
		Low	Up	Down	No count	Down
Counting on tim_ti2 only x2 mode	0010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	0011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicates the mechanical zero position, can be connected to the external trigger input and trigger a counter reset.

Figure 184 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are

selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S = 01 (TIMx_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIMx_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P and CC1NP = 0 (TIMx_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P and CC2NP = 0 (TIMx_CCER register, tim_ti2fp2 noninverted, tim_ti2fp2 = tim_ti2).
- SMS = 0011 (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIMx_CR1 register, counter is enabled).

Figure 184. Example of counter operation in encoder interface mode

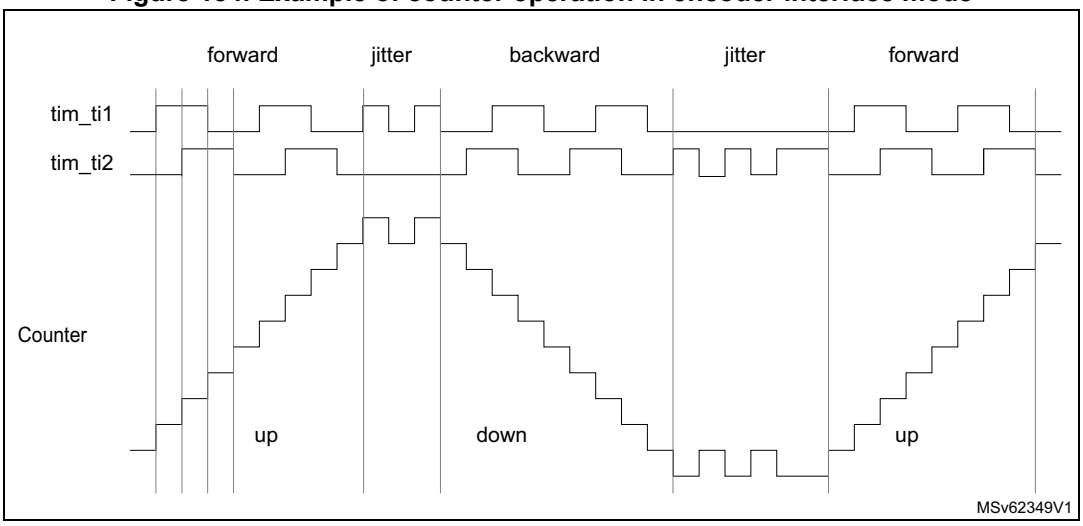


Figure 185 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 185. Example of encoder interface mode with tim_ti1fp1 polarity inverted

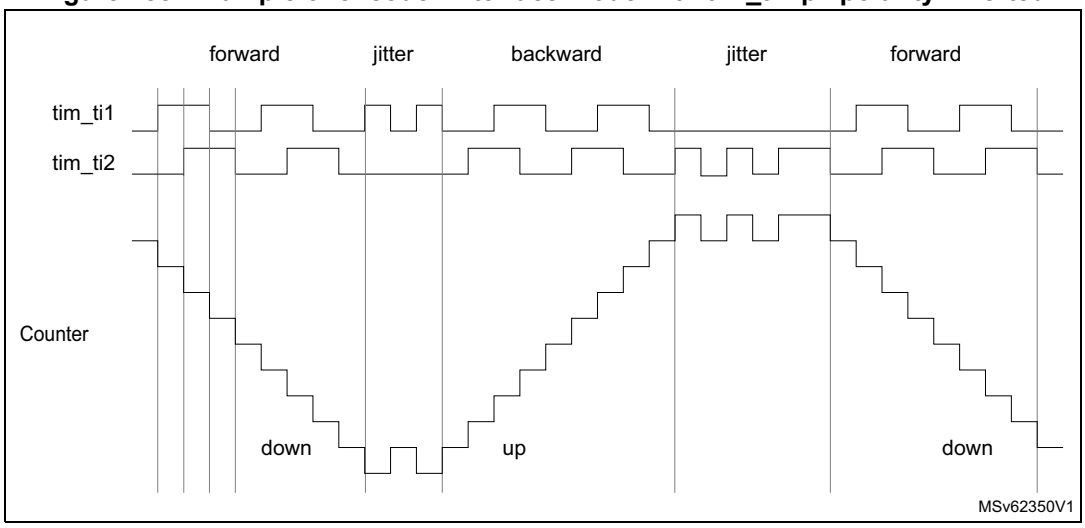
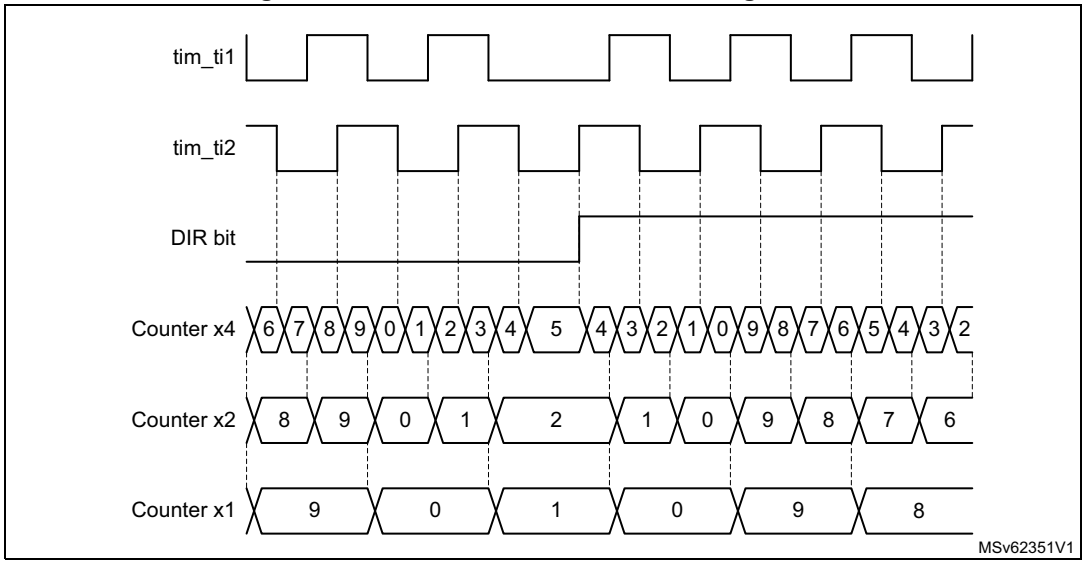


Figure 186 shows the timer counter value during a speed reversal, for various counting modes.

Figure 186. Quadrature encoder counting modes



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMx_CNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

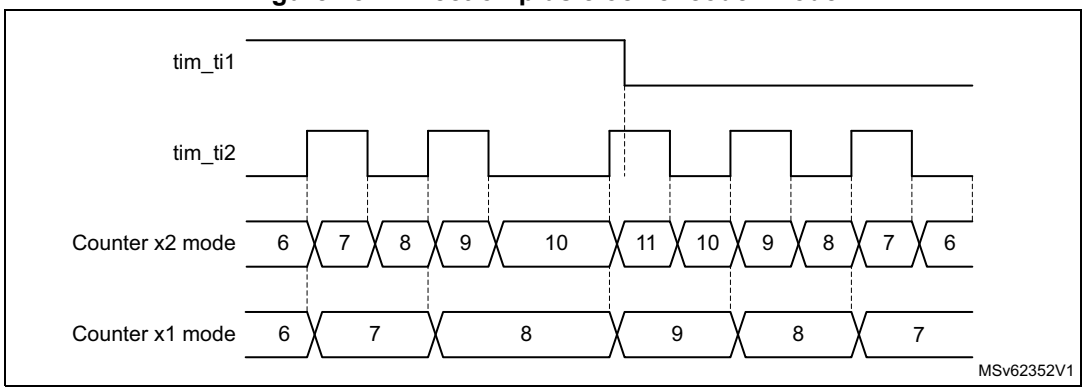
In the "clock plus direction" mode shown on *Figure 187*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1010: x2 mode, the counter is updated on both rising and falling edges of the clock.
- 1011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity.

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 187. Direction plus clock encoder mode



Directional clock encoder mode

In the “directional clock” mode on *Figure 188*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock lines. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 188*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 189*).
- 1101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 188*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 189*).

Figure 188. Directional clock encoder mode (CC1P = CC2P = 0)

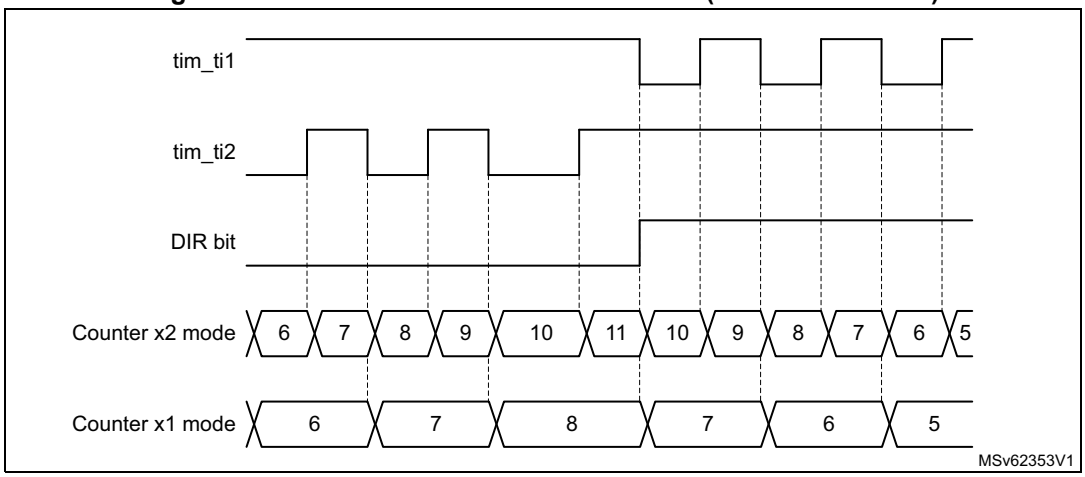


Figure 189. Directional clock encoder mode (CC1P = CC2P = 1)

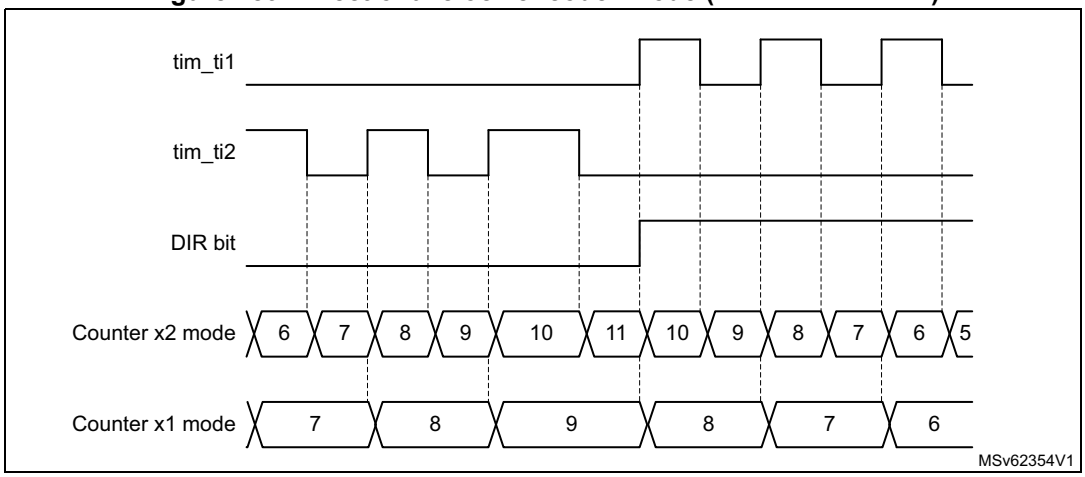


Table 261 details how the directional clock mode operates, for any input transition.

Table 261. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP = 0	1100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP = 1	1100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP = 0	1101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP = 1	1101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index input

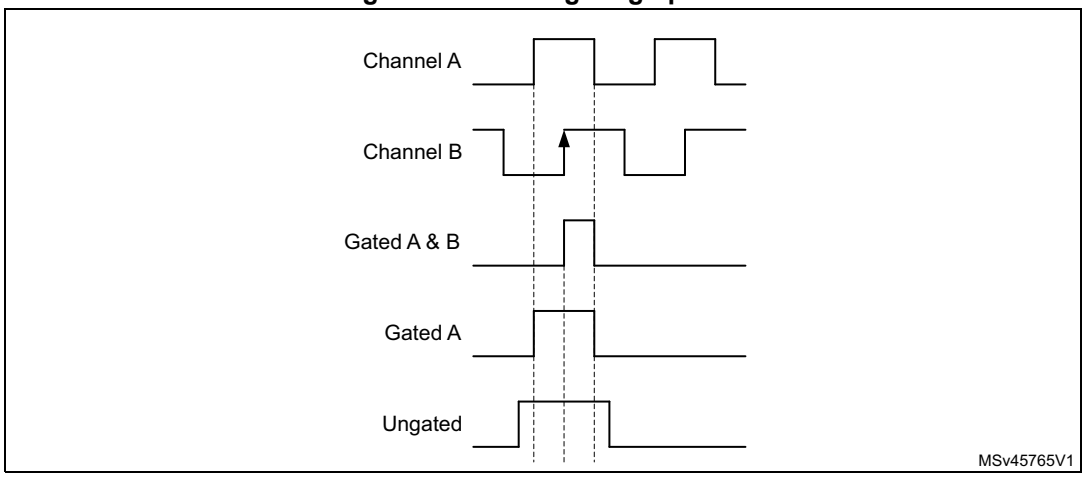
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIMx_ECR register. The IE bit must be set only in encoder mode, when the SMS[3:0] bitfield has the following values: 0001, 0010, 011, 1010, 1011, 1100, 1101, 1110, 1111.

Available encoders are proposed with several options for index pulse conditioning, as per *Figure 190*:

- Gated with A and B: the pulse width is 1/4 of one channel period, aligned with both A and B edges.
- Gated with A (or gated with B): the pulse width is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B).
- Ungated: the pulse width is up to one channel period, without any alignment to the edges.

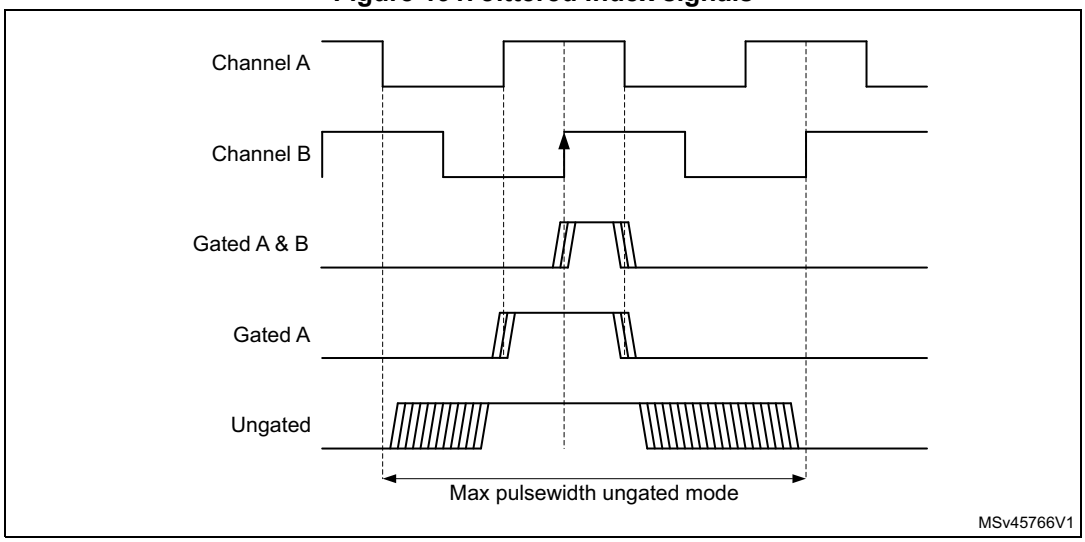
Figure 190. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as shown on *Figure 191*.

In ungated mode, the signal must be strictly below two encoder periods. If the pulse width is greater or equal to two encoder period, the counter is reset multiple times.

Figure 191. Jittered Index signals



The timer supports the three gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (for example channel A and channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIMx_ECR register.

The index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIMx_ARR when down-counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. *Figure 192* shows at which position is the index generated, for a simplistic example (an encoder providing four edges par mechanical rotation).

Figure 192. Index generation for IPOS[1:0] = 11

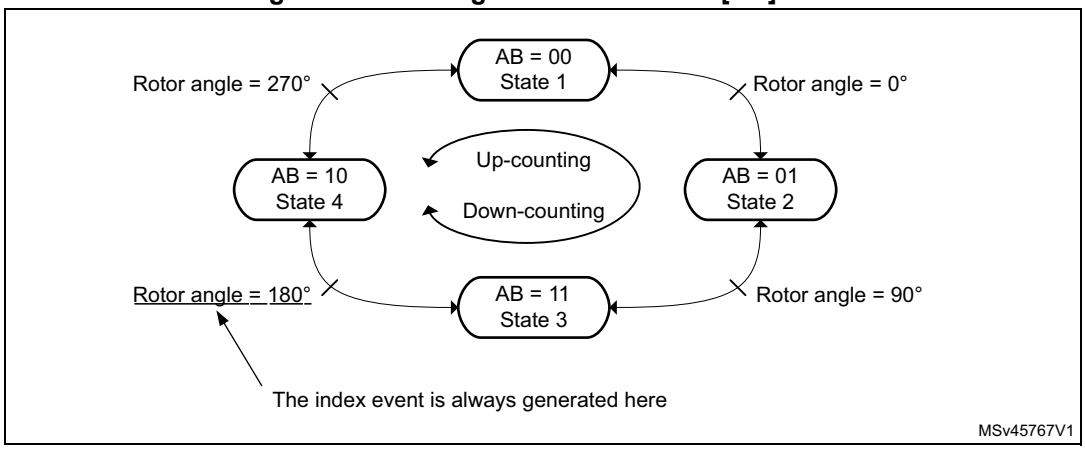


Figure 193 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is 11 (ChA = 1, ChB = 1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the 11 state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

Figure 193. Counter reading with index gated on channel A (IPOS[1:0] = 11)

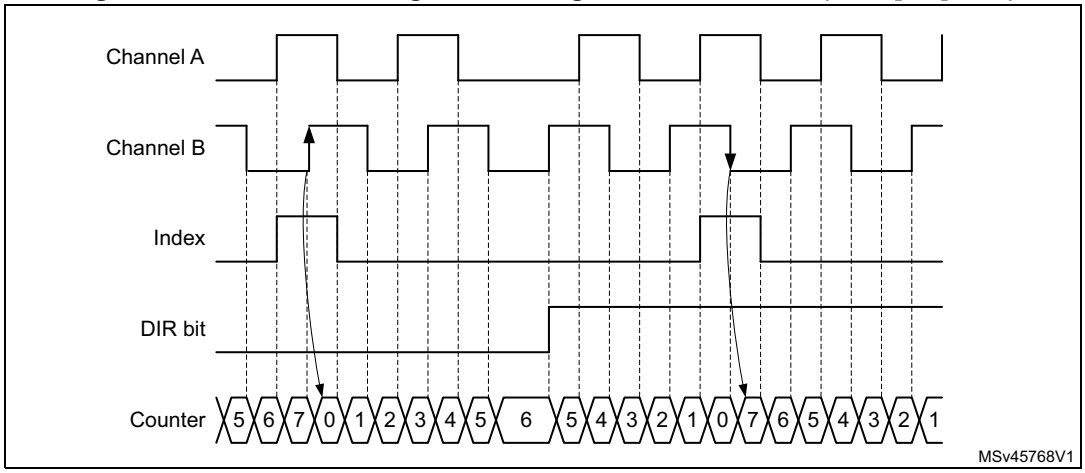


Figure 194 presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 194. Counter reading with index ungated (IPOS[1:0] = 00)

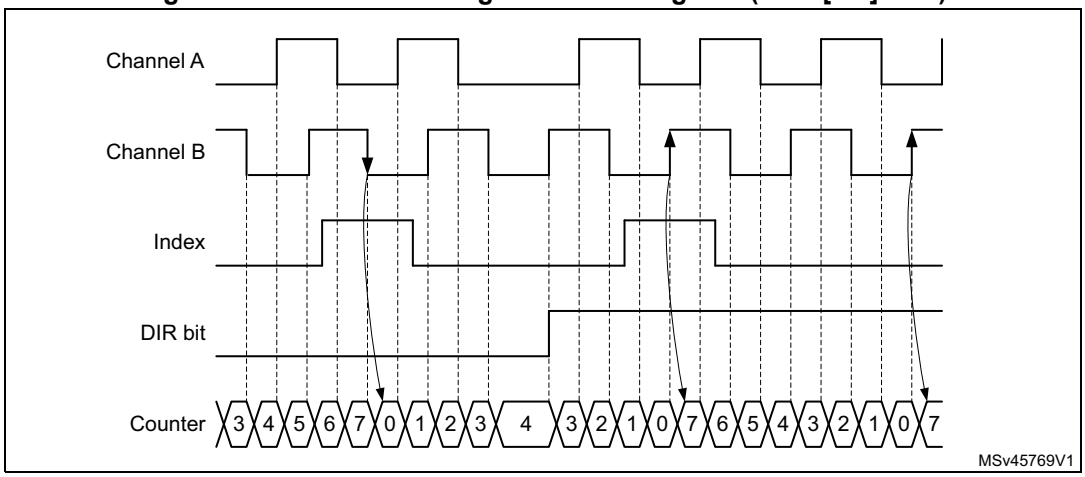


Figure 195 shows how the gated on A & B mode is handled, for various pulse alignment scenarios. The arrows are indicating on which transition is the index event generated.

Figure 195. Counter reading with index gated on channel A and B

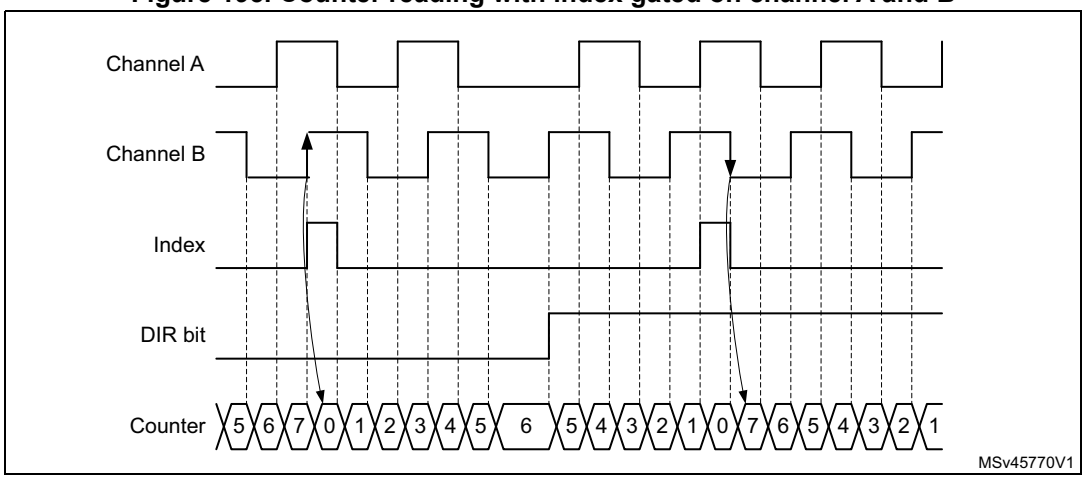


Figure 196 and *Figure 197* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 196. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

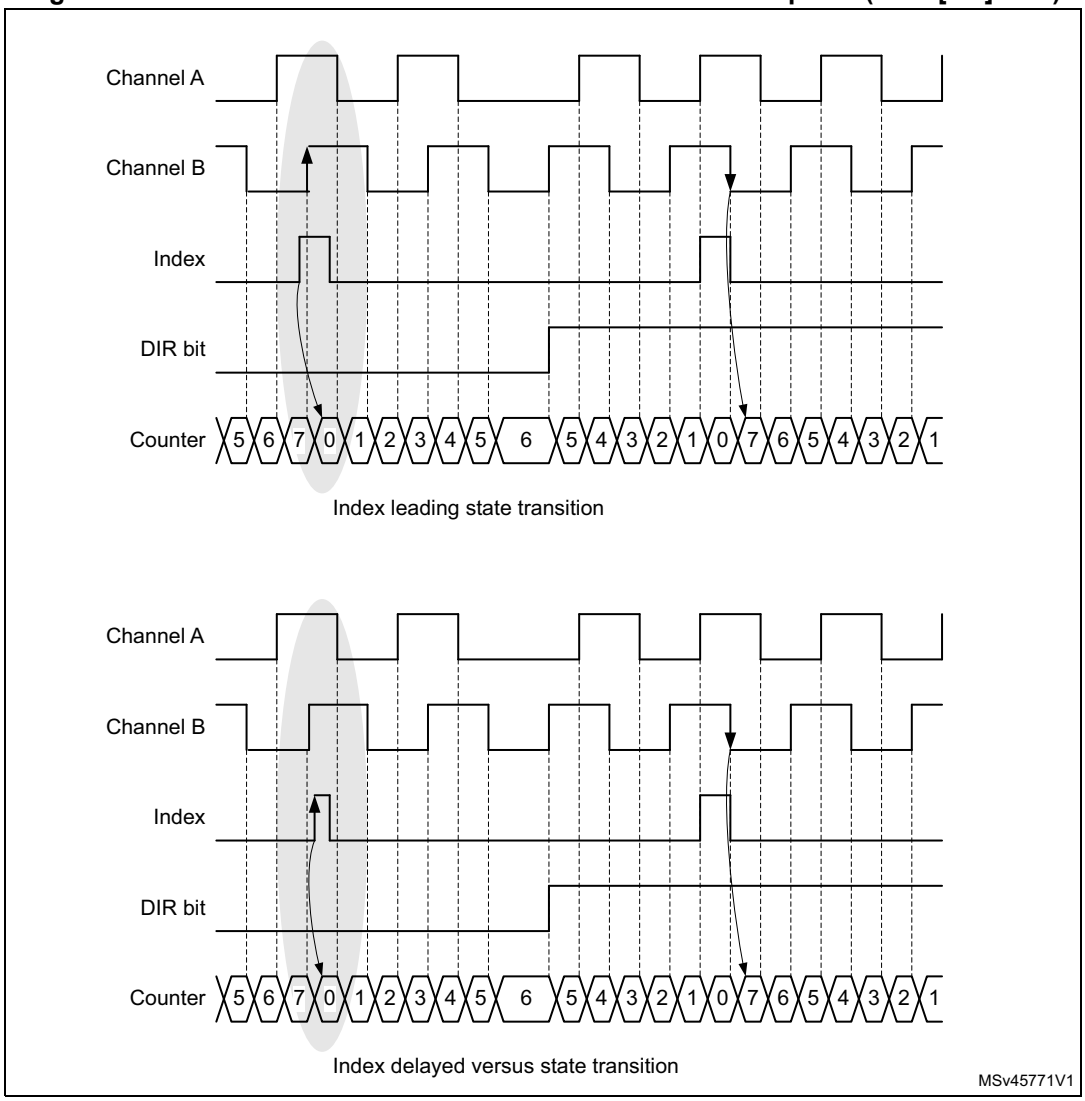


Figure 197. Counter reset Narrow index pulse (closer view, ARR = 0x07)

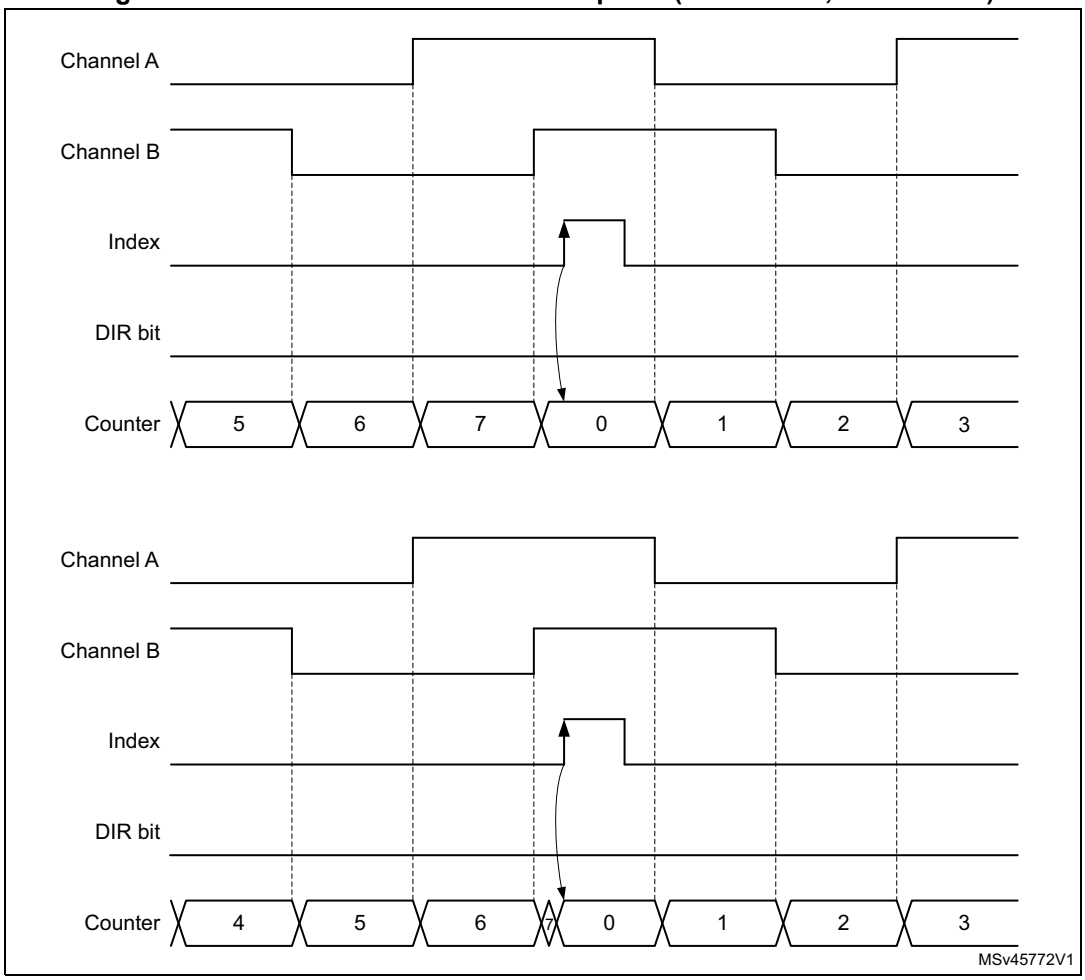
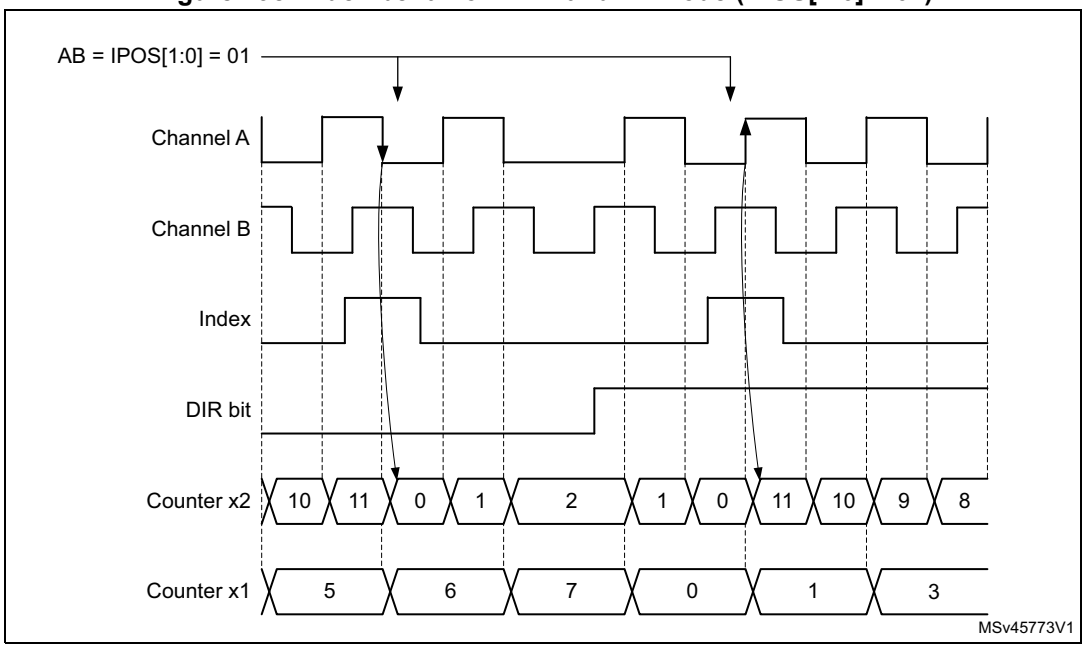


Figure 198 shows how the index is managed in x1 and x2 modes.

Figure 198. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)

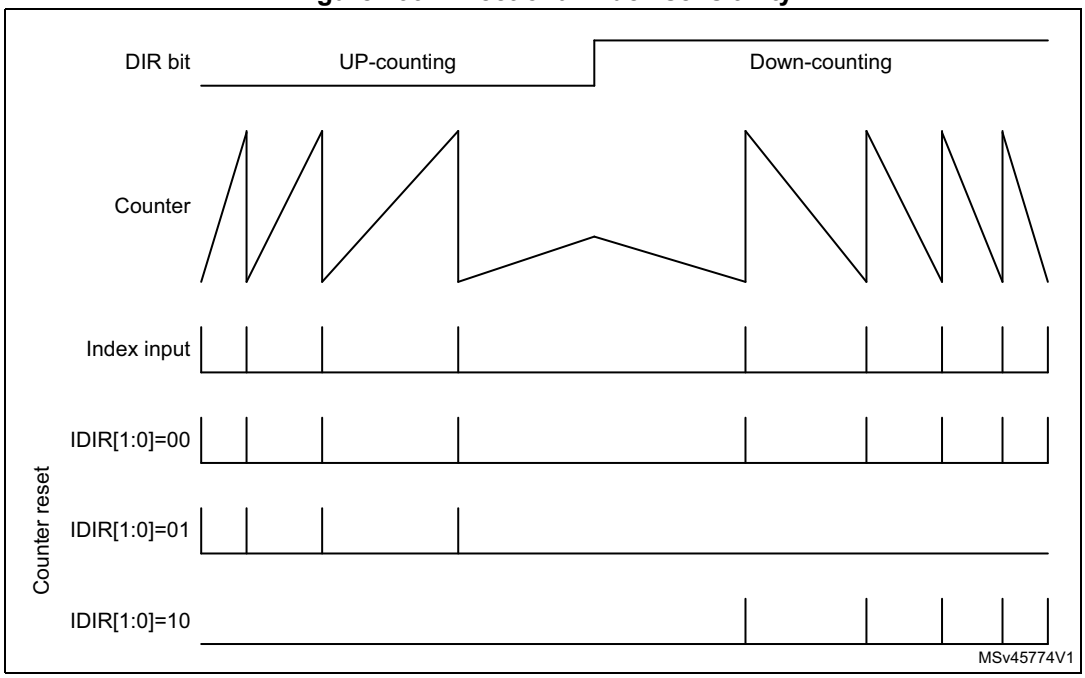


Directional index sensitivity

The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

Figure 199 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

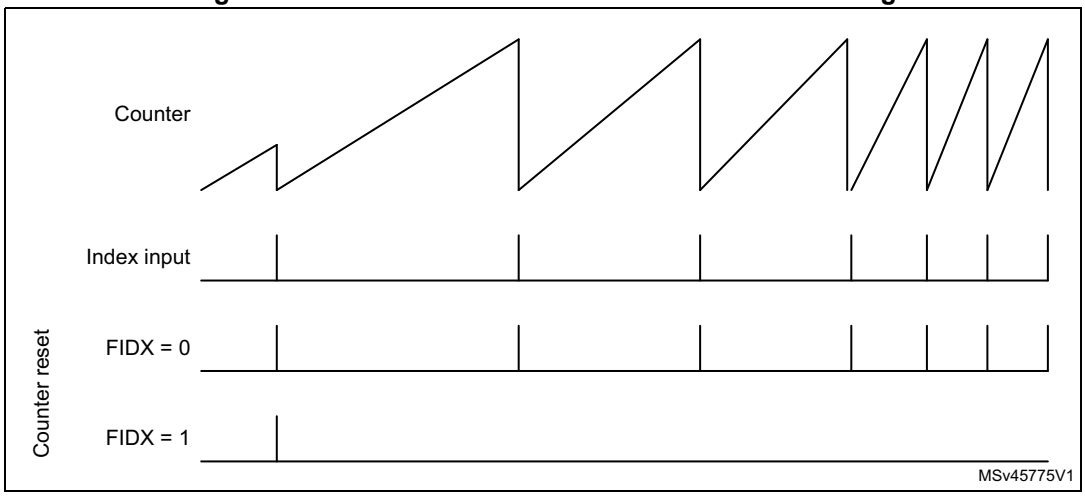
Figure 199. Directional index sensitivity



Special first index event management

The FIDX bit in the TIMx_ECR register allows the index to be taken only once, as shown on *Figure 200*. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be rearmed by writing the FIDX bit to 0 and setting it again to 1.

Figure 200. Counter reset as function of FIDX bit setting



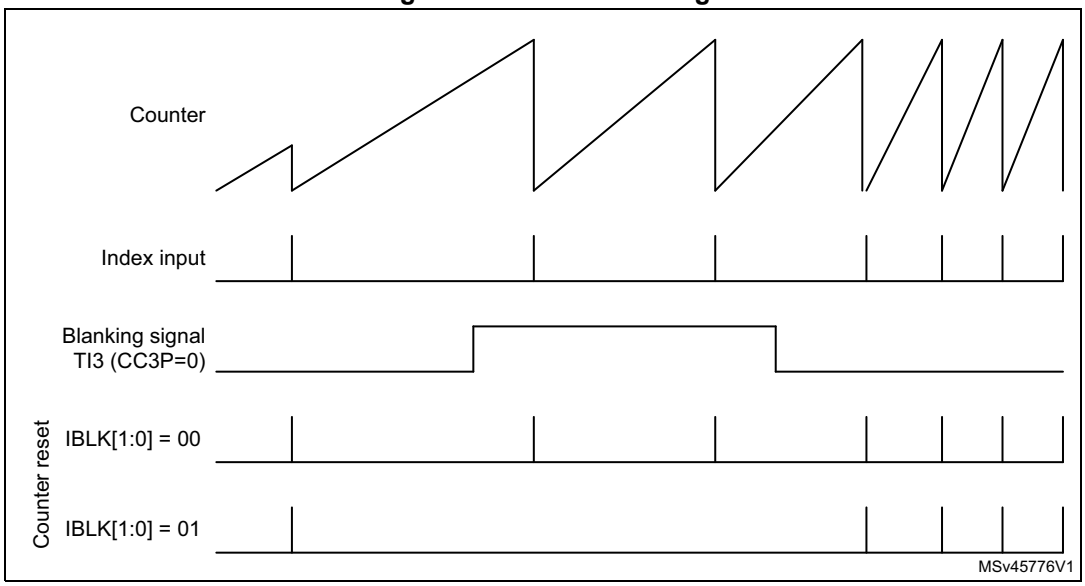
Index blanking

The index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on *Figure 201*.

This mode is enabled using the IBLK[1:0] bitfield in the TIMx_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active.
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input.
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input.

Figure 201. Index blanking



Index management in nonquadrature mode

Figure 202 and *Figure 203* detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[3:0] bitfield is equal to 1010, 1011, 1100, 1101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level.
- IPOS[0] = 1: Index is detected on clock high level.

The IPOS[1] bit is not-significant.

Figure 202. Index behavior in clock + direction mode, IPOS[0] = 1

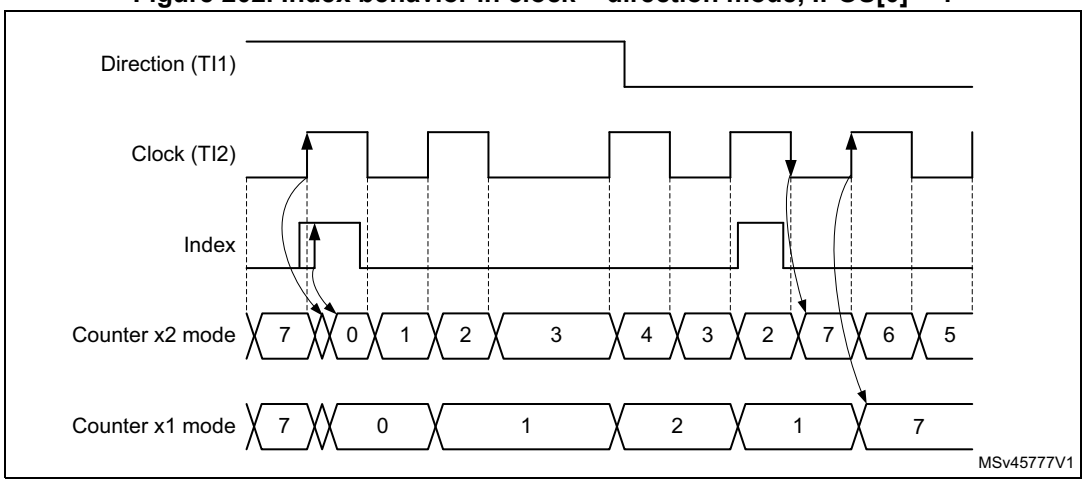
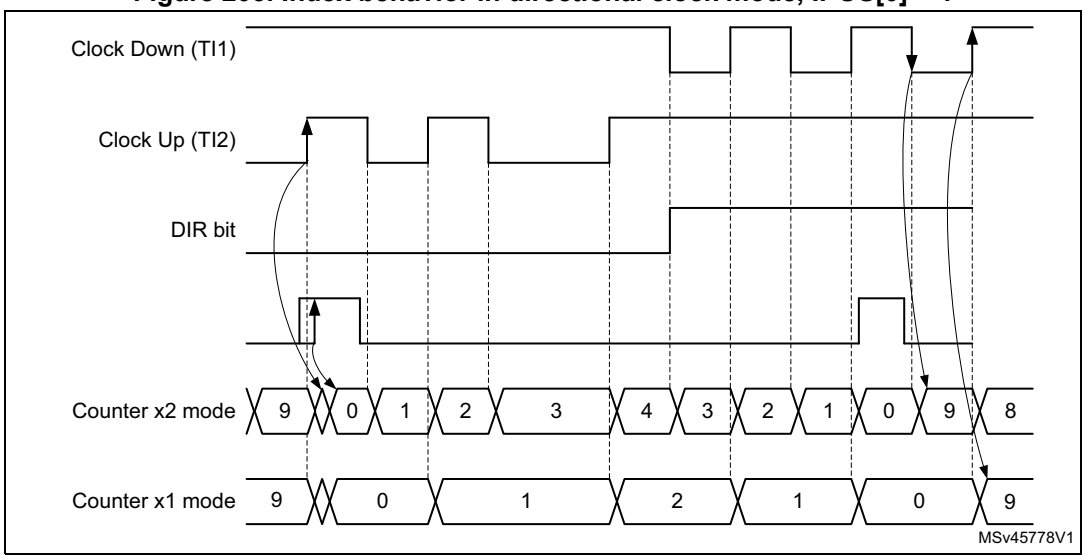


Figure 203. Index behavior in directional clock mode, IPOS[0] = 1

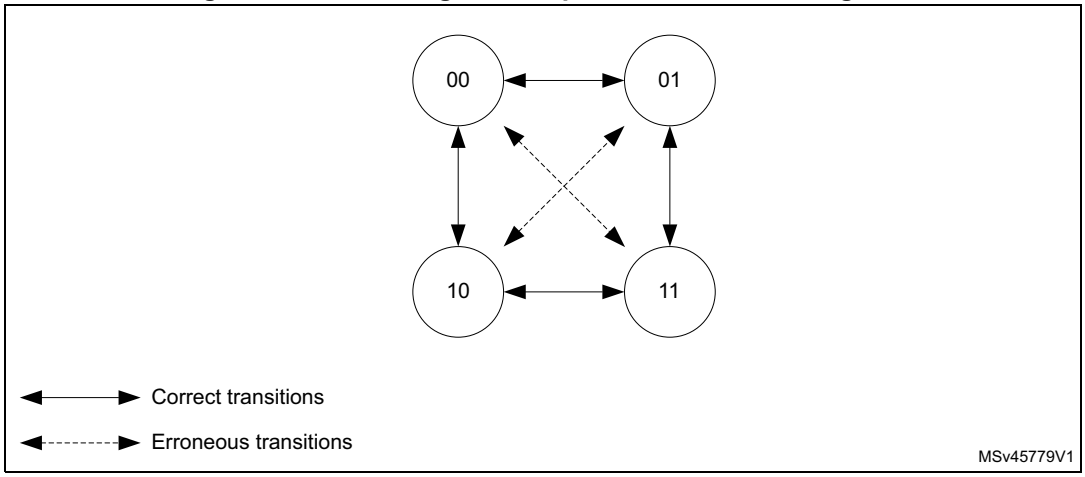


Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 204*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIMx_SR

status register. A transition error interrupt is generated if the TERRIE bit is set in the TIMx_DIER register.

Figure 204. State diagram for quadrature encoded signals



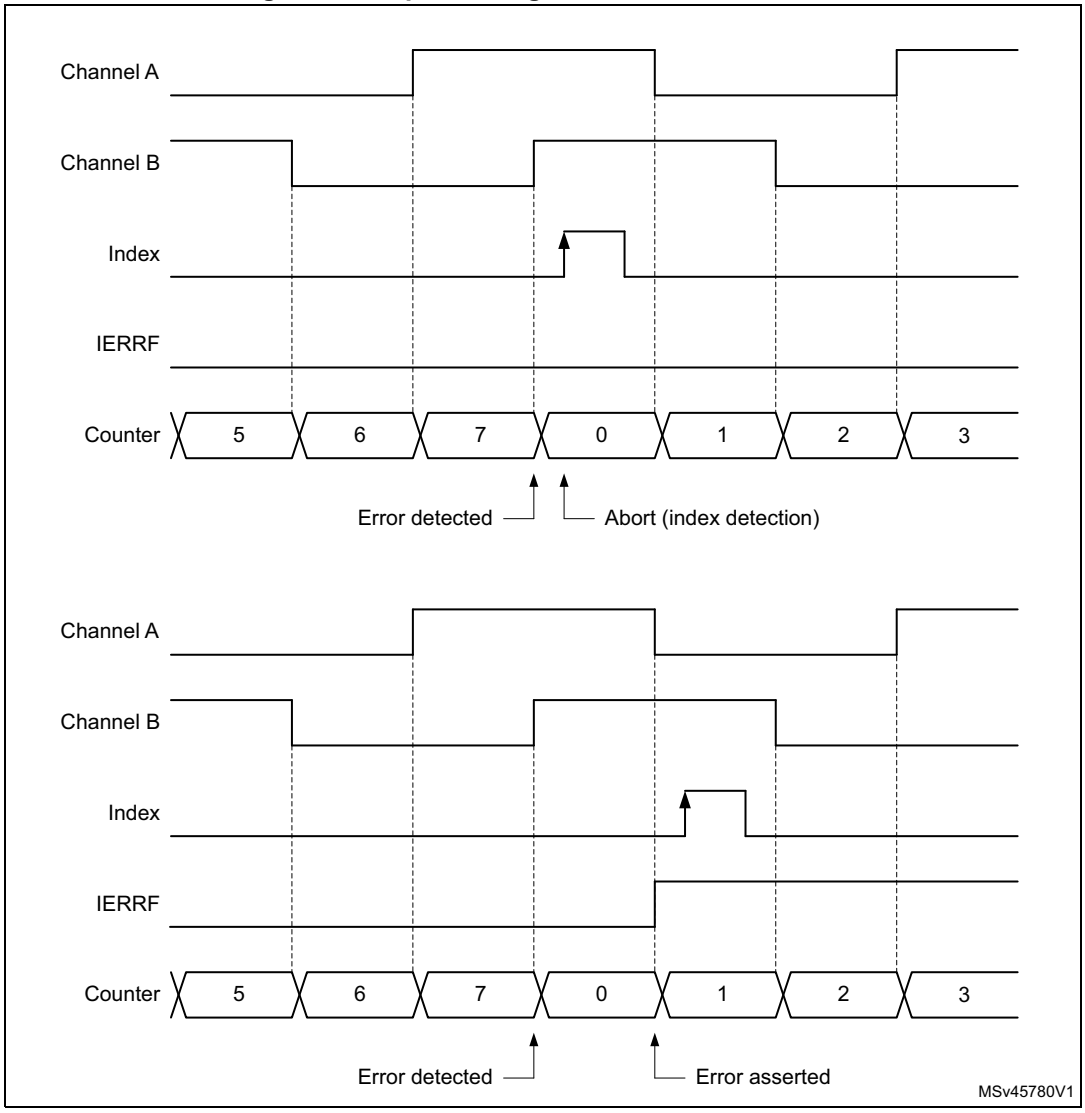
For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMx_ARR value without any index event, this is reported as an index position error.

The overflow threshold is programmed using the TIMx_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIMx_ARR = 3999 + 1 = 4000.

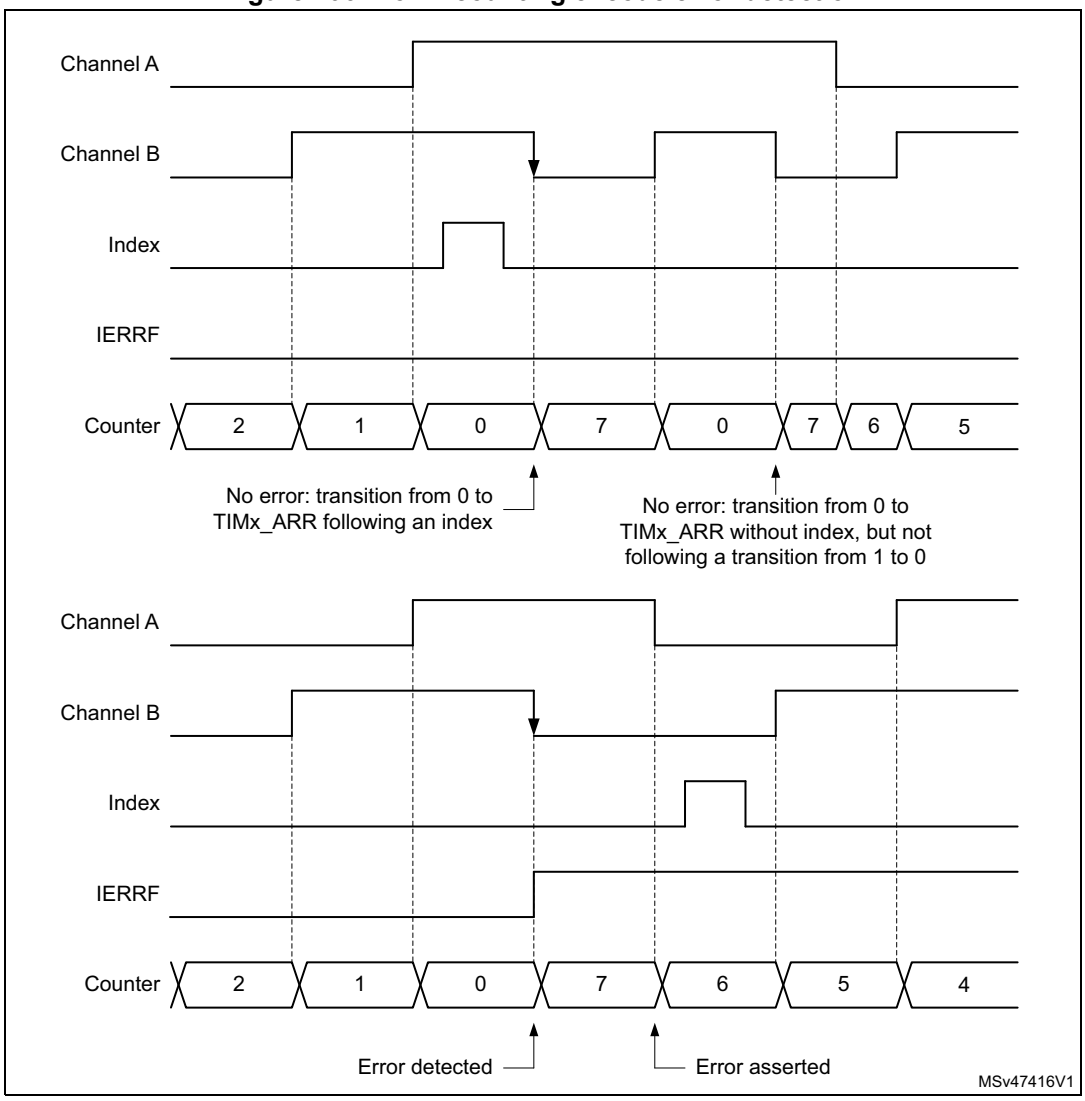
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 205*.

Figure 205. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 206*, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 206. Down-counting encode error detection



An index error sets the IERRF interrupt flag in the TIMx_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIMx_DIER register.

Functional encoder interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIMx_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIMx_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIMx_DIER register.
- Index event: the index event sets the IDXF interrupt flag in the TIMx_SR status register. An index interrupt is generated if the IDXIE bit is set in the TIMx_DIER register.

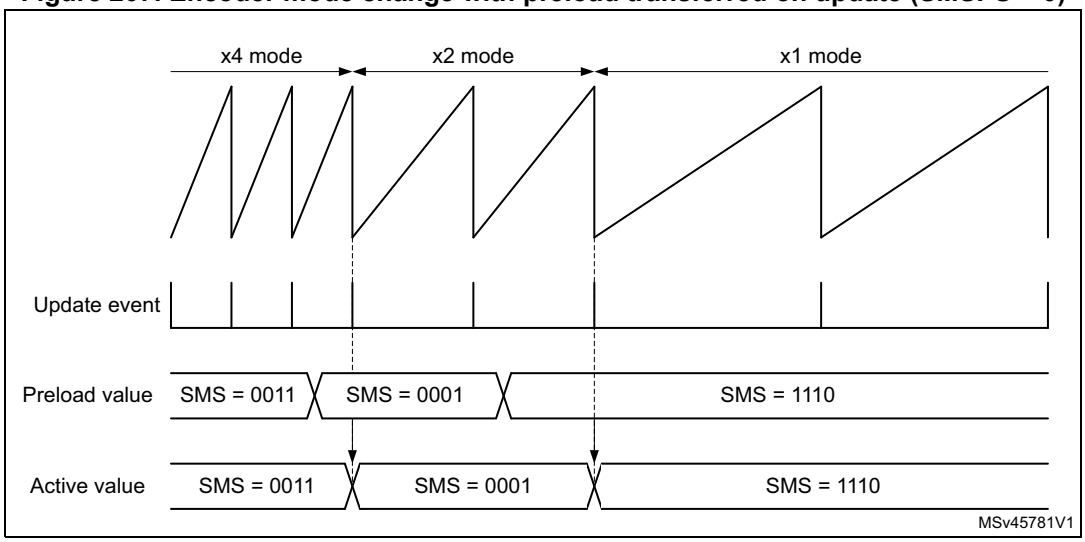
Slave mode selection preload for run-time encoder mode update

It can be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on *Figure 207*.

For this purpose, the SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value can be selected with the SMSPS bit in the TIMx_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when up-counting, and underflows when down-counting.
- SMSPS = 1: the transfer is triggered by the index event.

Figure 207. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIMx_CR2 register. It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

28.4.19 Direction bit output

It is possible to output a direction signal out of the timer, on the tim_oc3 and tim_oc4 output signals (copy of the DIR bit in the TIMx_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bitfield to 1011 in the TIMx_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

28.4.20 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register's bit 31 (TIMx_CNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

28.4.21 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

28.4.22 Timers and external trigger synchronization

The TIMx timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

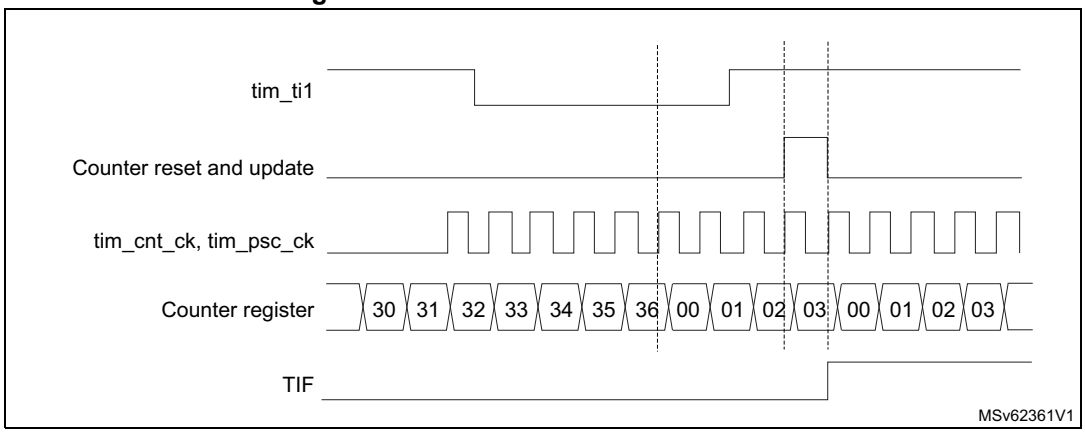
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Start the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the autoreload register TIMx_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 208. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

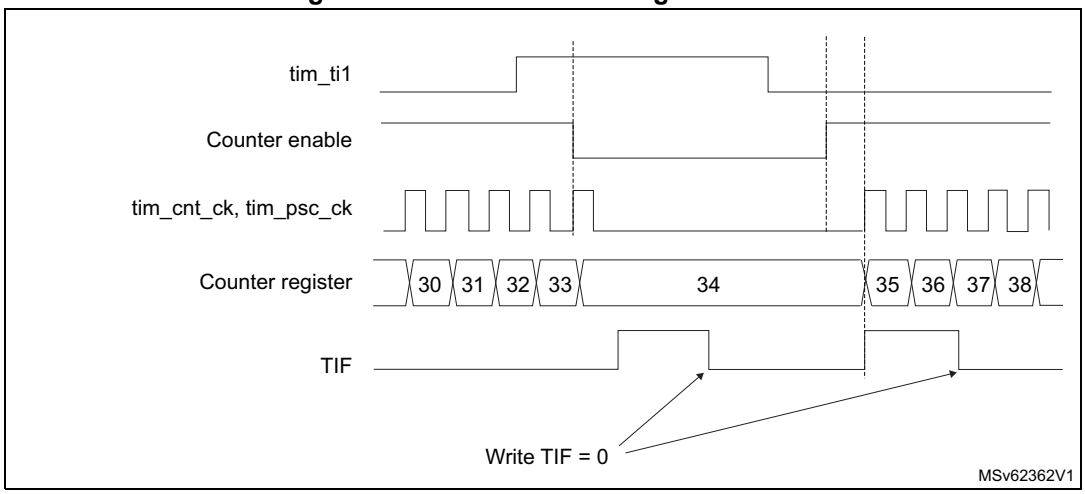
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 209. Control circuit in gated mode



Note: The configuration "CCxP = CCxNP = 1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

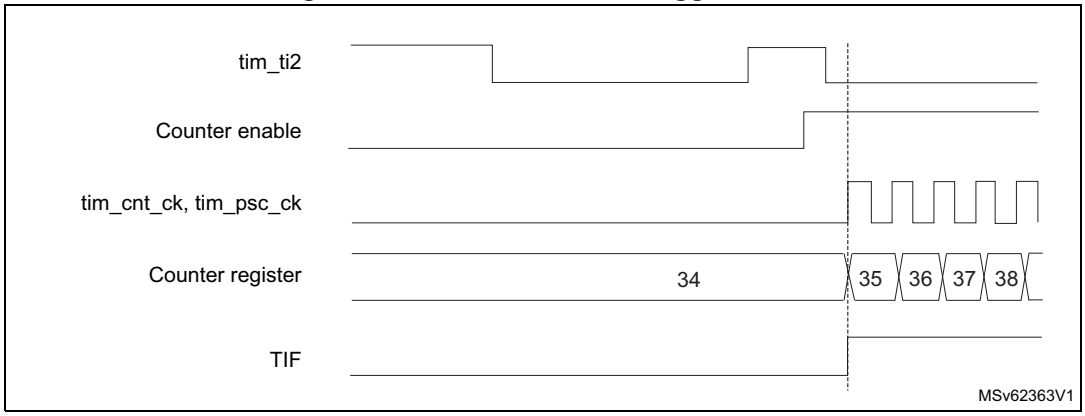
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S = 01 in TIMx_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 210. Control circuit in trigger mode



Slave mode selection preload for run-time encoder mode update

The SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value is the update event (UEV) occurring when the counter overflows.

Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode – combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode – external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode, or trigger mode. It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIMx_SMCR register.

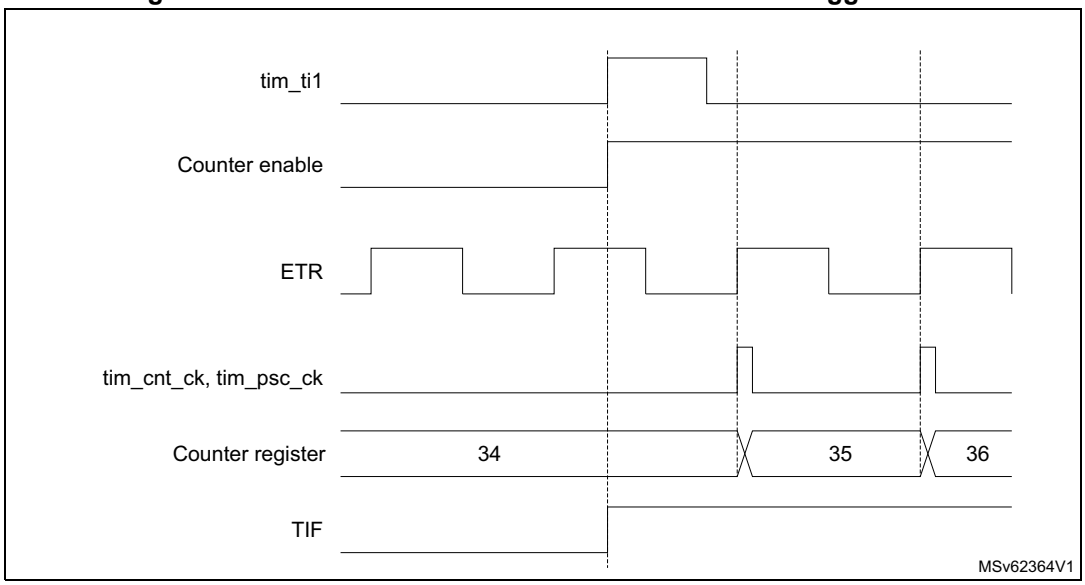
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter.
   - ETPS = 00: prescaler disabled.
   - ETP = 0: detection of rising edges on tim_etr_in and ECE = 1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01in TIMx_CCMR1 register to select only the input capture source.
   - CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 211. Control circuit in external clock mode 2 + trigger mode



28.4.23 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one timer is configured in Master mode, it can reset, start, stop, or clock the counter of another timer configured in Slave mode.

Figure 212 and *Figure 213* show examples of master/slave timer connections.

Figure 212. Master/Slave timer example

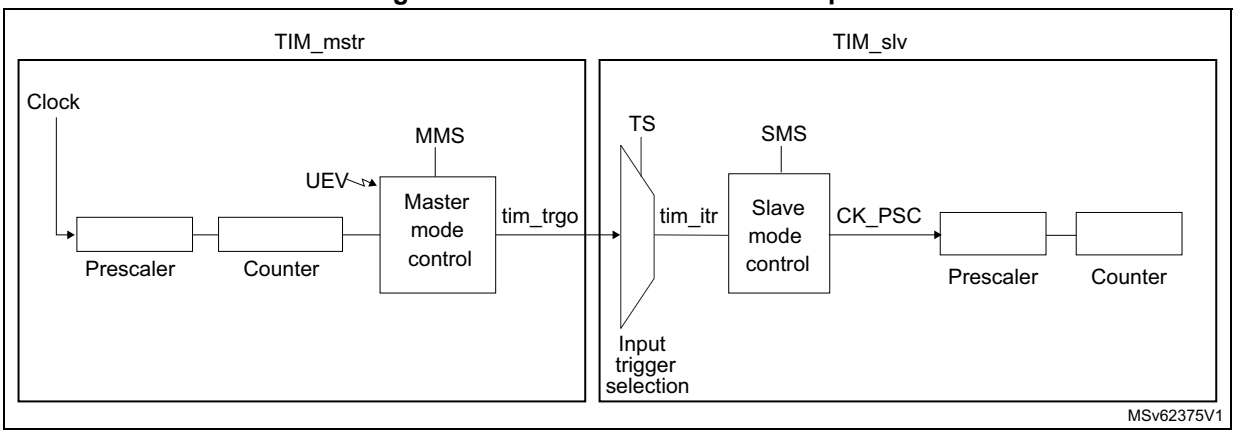
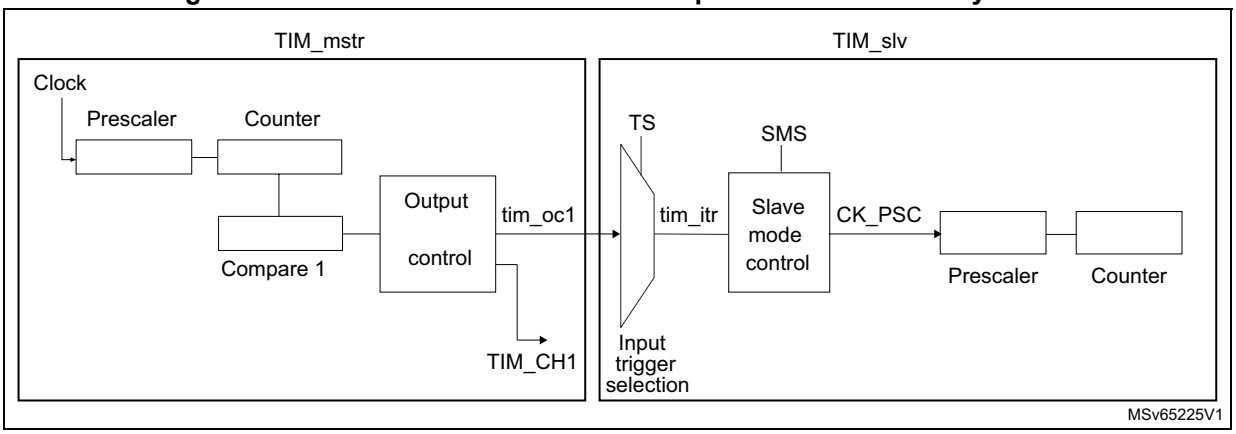


Figure 213. Master/slave connection example with 1 channel only timers



Note: The timers with one channel only (see Figure 213) do not feature a master mode. However, the tim_oc1 output signal can serve as trigger for slave timer (see TIMx internal trigger connection table in Section 28.4.2: TIM2 pins and internal signals).
The tim_oc1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.
For instance, if the destination timer tim_ker_ck clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

Using one timer as prescaler for another timer

For example, TIM_mstr can be configured to act as a prescaler for TIM_slv. Refer to *Figure 212*. To do this:

1. Configure TIM_mstr in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS = 010 is written in the TIM_mstr_CR2 register, a rising edge is output on tim_trgo each time an update event is generated.
2. To connect the tim_trgo output of TIM_mstr to TIM_slv, TIM_slv must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM_slv_SMCR register (writing TS = 00010).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS = 111 in the TIM_slv_SMCR register). This causes TIM_slv to be clocked by the rising edge of the periodic TIM_mstr trigger signal (which correspond to the TIM_mstr counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If tim_ocx is selected on TIM_mstr as the trigger output (MMS = 1xx), its rising edge is used to clock the counter of TIM_slv.

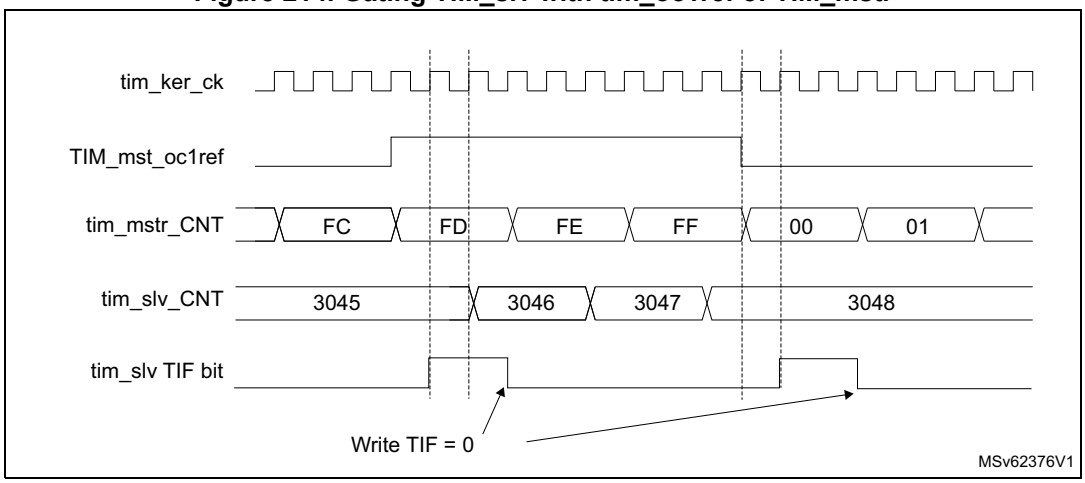
Using one timer to enable another timer

In this example, we control the enable of TIM_slv with the output compare 1 of TIM_mstr. Refer to *Figure 212* for connections. TIM_slv counts on the divided internal clock only when tim_oc1ref of TIM_mstr is high. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim_cnt_ck} = f_{tim_ker_ck}/3$).

1. Configure TIM_mstr master mode to send its output compare 1 reference (tim_oc1ref) signal as trigger output (MMS = 100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS = 101 in TIM_slv_SMCR register).
5. Enable TIM_slv by writing 1 in the CEN bit (TIM_slv_CR1 register).
6. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).

Note: The slave timer counter clock is not synchronized with the master timer counter clock, this mode only affects the TIM_slv counter enable signal.

Figure 214. Gating TIM_slv with tim_oc1ref of TIM_mstr

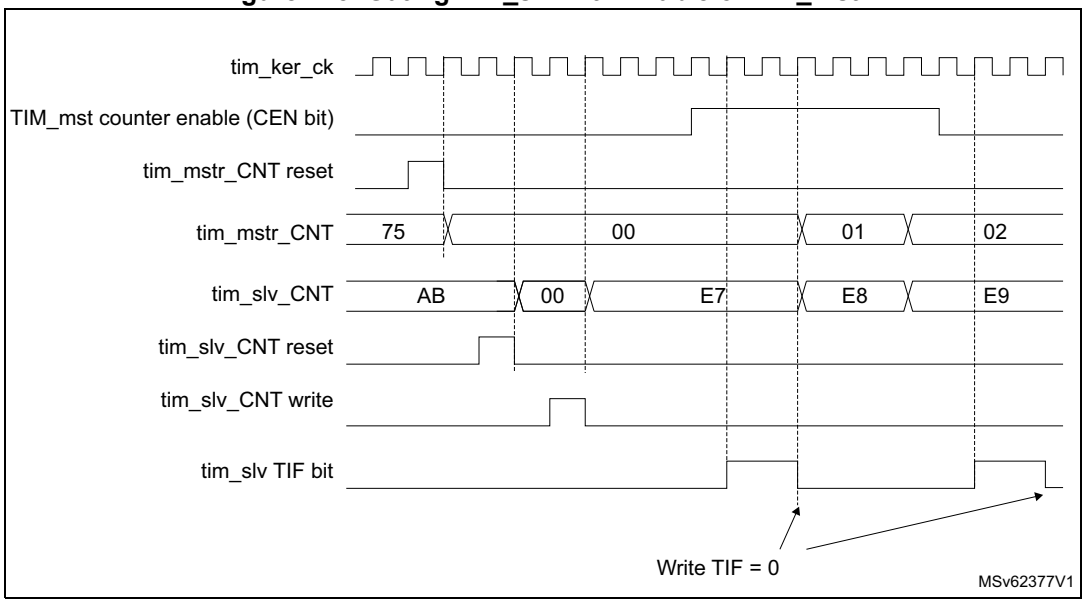


In the example in *Figure 214*, the TIM_slv counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM_mstr. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to *Figure 215*), we synchronize TIM_mstr and TIM_slv. TIM_mstr is the master and starts from 0. TIM_slv is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM_slv stops when TIM_mstr is disabled by writing 0 to the CEN bit in the TIM_mstr_CR1 register:

1. Configure TIM_mstr master mode to send its output compare 1 reference (tim_oc1ref) signal as trigger output (MMS = 100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS = 101 in TIM_slv_SMCR register).
5. Reset TIM_mstr by writing 1 in UG bit (TIM_mstr_EGR register).
6. Reset TIM_slv by writing 1 in UG bit (TIM_slv_EGR register).
7. Initialize TIM_slv to 0xE7 by writing 0xE7 in the TIM_slv counter (TIM_slv_CNT).
8. Enable TIM_slv by writing 1 in the CEN bit (TIM_slv_CR1 register).
9. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).
10. Stop TIM_mstr by writing 0 in the CEN bit (TIM_mstr_CR1 register).

Figure 215. Gating TIM_slv with Enable of TIM_mstr

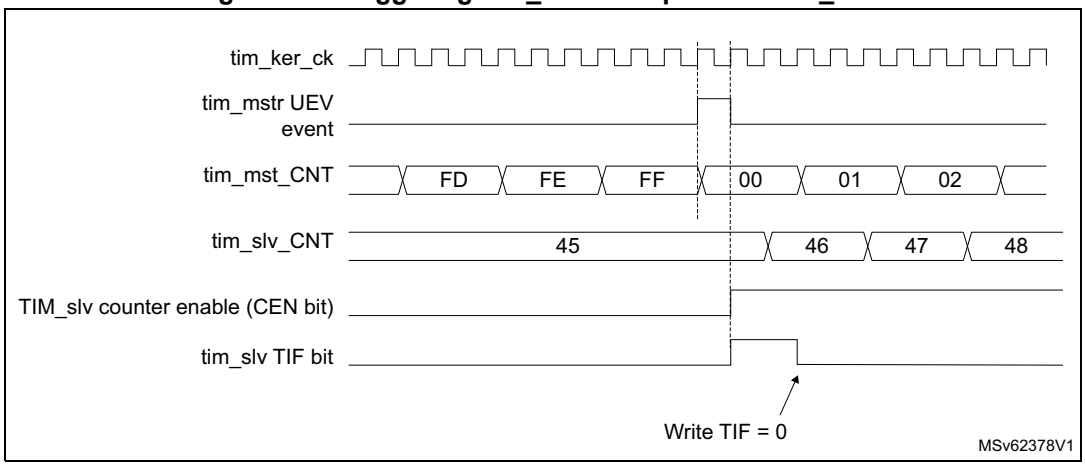


Using one timer to start another timer

In this example, we set the enable of TIM_slv with the update event of TIM_mstr. Refer to *Figure 212* for connections. TIM_slv starts counting from its current value (which can be nonzero) on the divided internal clock as soon as the update event is generated by TIM_mstr. When TIM_slv receives the trigger signal its CEN bit is automatically set and the counter counts until we write 0 to the CEN bit in the TIM_slv_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim_cnt_ck} = f_{tim_ker_ck}/3$).

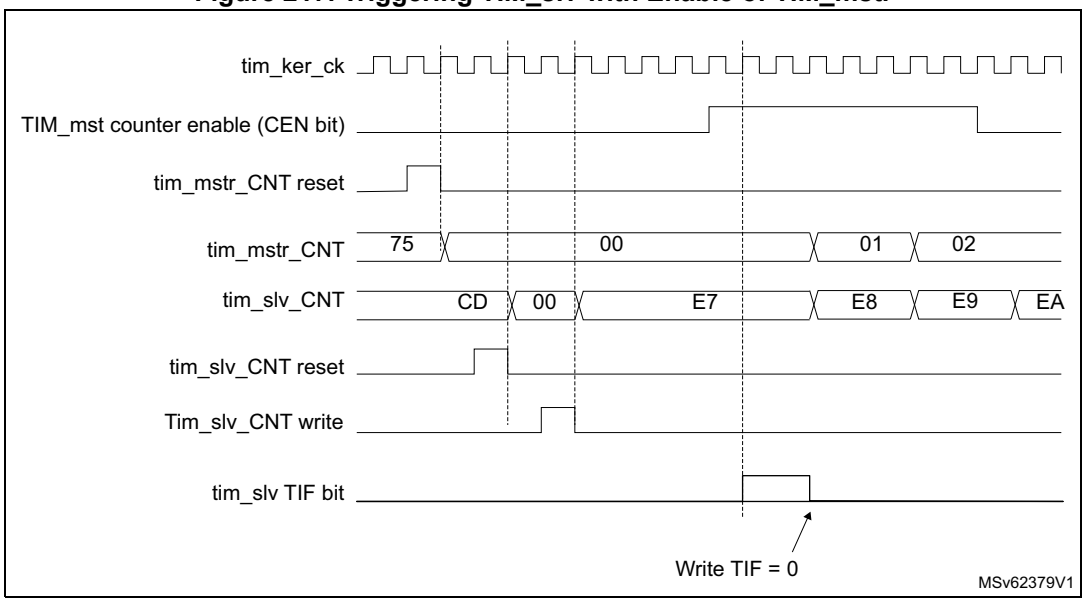
1. Configure TIM_mstr master mode to send its update event (UEV) as trigger output (MMS = 010 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr period (TIM_mstr_ARR registers).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in trigger mode (SMS = 110 in TIM_slv_SMCR register).
5. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).

Figure 216. Triggering TIM_slv with update of TIM_mstr



As in the previous example, both counters can be initialized before starting counting. *Figure 217* shows the behavior with the same configuration as in *Figure 216* but in trigger mode (SMS = 110 in the TIM_slv_SMCR register) instead of gated mode.

Figure 217. Triggering TIM_slv with Enable of TIM_mstr



Starting two timers synchronously in response to an external trigger

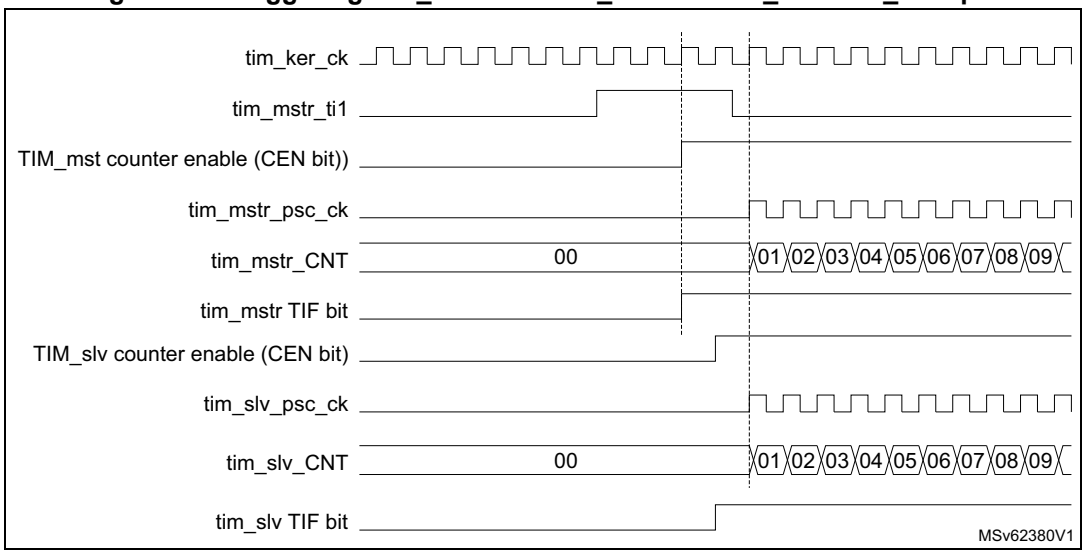
In this example, we set the enable of TIM_mstr when its tim_ti1 input rises, and the enable of TIM_slv with the enable of TIM_mstr. Refer to *Figure 212* for connections. To ensure the counters are aligned, TIM_mstr must be configured in Master/Slave mode (slave with respect to tim_ti1, master with respect to TIM_slv):

1. Configure TIM_mstr master mode to send its enable as trigger output (MMS = 001 in the TIM_mstr_CR2 register).
2. Configure TIM_mstr slave mode to get the input trigger from tim_ti1 (TS = 00100 in the TIM_mstr_SMCR register).
3. Configure TIM_mstr in trigger mode (SMS = 110 in the TIM_mstr_SMCR register).
4. Configure the TIM_mstr in Master/Slave mode by writing MSM = 1 (TIM_mstr_SMCR register).
5. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00000 in the TIM_slv_SMCR register).
6. Configure TIM_slv in trigger mode (SMS = 110 in the TIM_slv_SMCR register).

When a rising edge occurs on tim_ti1 (TIM_mstr), both counters start counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIMx_CNT). One can see that the master/slave mode inserts a delay between CNT_EN and CK_PSC on TIM_mstr.

Figure 218. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input



Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

28.4.24 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

28.4.25 ADC synchronization

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIMx_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- When the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.
- When the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source):

1. The counter period must be a multiple of the ADC clock period.

$$(\text{TIMx_PSC} + 1) \times (\text{TIMx_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period.

$$(\text{TIMx_PSC} + 1) \times \text{TIMx_CMPy} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements is met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock.
2. Configure the timer and set the ADSYNC bit.
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits).
4. Start the timer (with the CEN counter enable bit).

28.4.26 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to reprogram part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write accesses are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 28.5.23: TIM2 DMA control register (TIM2_DCR)* for details).

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address.
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bitfields as follows: DBL = 3 transfers, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx.
5. Enable the DMA channel.

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

28.4.27 TIM2 DMA requests

The TIM2 can generate a DMA requests, as shown in *Table 262*.

Table 262. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_cc3_dma	Capture/compare 3	CC3DE
tim_cc4_dma	Capture/compare 4	CC4DE
tim_trgi_dma	Trigger	TDE

Note: Some timer's DMA requests may not be connected to the DMA controller. Refer to the DMA section(s) for more details.

28.4.28 Debug mode

When the microcontroller enters debug mode (Core core halted), the TIMx counter can either continue to work normally or stops.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

28.4.29 TIM2 low-power modes

Table 263. Effect of low-power modes on TIM2

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop 0 and 1	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Stop 2, 3, and Standby	The timer is powered-down and must be reinitialized after exiting the Stop 2, 3, and Standby modes.

28.4.30 TIM2 interrupts

The TIM2 can generate multiple interrupts, as shown in *Table 264*.

Table 264. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UP	Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Capture/compare 3	CC3IF	CC3IE	write 0 in CC3IF	Yes	No
	Capture/compare 4	CC4IF	CC4IE	write 0 in CC4IF	Yes	No
TIM_TRG	Trigger	TIF	TIE	write 0 in TIF	Yes	No
TIM_DIR_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_IERR	Index Error	IERRF	IERRIE	write 0 in IERRF	Yes	No
TIM_TER	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

28.5 TIM2 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

28.5.1 TIM2 control register 1 (TIM2_CR1)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering Enable

0: Dithering disabled

1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.

1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_etr_in, tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$

01: $t_{DTS} = 2 \times t_{tim_ker_ck}$

10: $t_{DTS} = 4 \times t_{tim_ker_ck}$

11: Reserved

Bit 7 **ARPE**: Autoreload preload enable

0: TIMx_ARR register is not buffered

1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).

01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting down.

10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting up.

11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter

1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

28.5.2 TIM2 control register 2 (TIM2_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	AD SYNC	Res.	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw			rw									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ADSYNC**: ADC synchronization

0: The timer operates independently from the ADC

1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 28.4.25* for requirements.

The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bits 27:26 Reserved, must be kept at reset value.

Bits 24:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15:0] multiplexer output is to tim_ti1 input

1: The tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed and connected to the tim_ti1 input.

Bits 25, 6, 5, 4 **MMS[3:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode.
When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

Others: Reserved

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bits 2:0 Reserved, must be kept at reset value.

28.5.3 TIM2 slave mode control register (TIM2_SMCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
						rw	rw			rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **SMSPS**: SMS preload source

This bit selects whether the events that triggers the SMS[3:0] bitfield transfer from preload to active

0: The transfer is triggered by the Timer's Update event

1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded

0: SMS[3:0] bitfield is not preloaded

1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether tim_etr_in or $\overline{\text{tim_etr_in}}$ is used for trigger operations

0: tim_etr_in is non-inverted, active at high level or rising edge

1: tim_etr_in is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS = 111 and TS = 00111).
It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).
If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of tim_ker_ck frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF

01: tim_etrp frequency divided by 2

10: tim_etrp frequency divided by 4

11: tim_etrp frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bitfield then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bit 7 **MSM**: Master/Slave mode

0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]**: Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal trigger 0 (tim_itr0)
00001: Internal trigger 1 (tim_itr1)
00010: Internal trigger 2 (tim_itr2)
00011: Internal trigger 3 (tim_itr3)
00100: tim_ti1 edge detector (tim_ti1f_ed)
00101: Filtered timer input 1 (tim_ti1fp1)
00110: Filtered timer input 2 (tim_ti2fp2)
00111: External trigger input (tim_etrf)
01000: Internal trigger 4 (tim_itr4)
01001: Internal trigger 5 (tim_itr5)
01010: Internal trigger 6 (tim_itr6)
01011: Internal trigger 7 (tim_itr7)
01100: Internal trigger 8 (tim_itr8)
01101: Internal trigger 9 (tim_itr9)
01110: Internal trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
Others: Reserved

See *Section 28.4.2: TIM2 pins and internal signals* for product specific implementation details.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS**: OCREF clear selection

This bit is used to select the OCREF clear source
0: tim_ocref_clr_int is connected to the tim_ocref_clr input
1: tim_ocref_clr_int is connected to tim_etrf

Note: If the OCREF clear selection feature is not supported, this bit is reserved and forced by hardware to 0. Section 28.3: TIM2 implementation.

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001: Encoder mode 1 - Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

0010: Encoder mode 2 - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

0011: Encoder mode 3 - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

1010: Encoder mode: Clock plus direction, x2 mode.

1011: Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P.

1100: Encoder mode: Directional clock, x2 mode.

1101: Encoder mode: Directional clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

1110: Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

1111: Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

28.5.4 TIM2 DMA/Interrupt enable register (TIM2_DIER)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable
0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction change interrupt disabled
1: Direction change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable
0: Index interrupt disabled
1: Index interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled.
1: Trigger DMA request enabled.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
0: CC4 DMA request disabled.
1: CC4 DMA request enabled.

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled.
1: CC3 DMA request enabled.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled.
1: CC2 DMA request enabled.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled.
1: CC1 DMA request enabled.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled.
1: CC4 interrupt enabled.

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled.
1: CC3 interrupt enabled.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

28.5.5 TIM2 status register (TIM2_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.
								rc_w0	rc_w0	rc_w0	rc_w0				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag
This flag is set by hardware when an index error is detected. It is cleared by software by writing it to 0.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIMx_CR is changing). It is cleared by software by writing it to 0.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to 0.
0: No index event occurred.
1: An index event has occurred

Bits 19:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input) when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag
Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag
Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are three possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
At overflow or underflow and if UDIS = 0 in the TIMx_CR1 register.
When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.
When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

28.5.6 TIM2 event generation register (TIM2_EGR)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation
Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (up-counting), else it takes the autoreload value (TIMx_ARR) if DIR = 1 (down-counting).

28.5.7 TIM2 capture/compare mode register 1 (TIM2_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1.

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2

0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4

0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8

0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6

0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8

0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6

0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8

1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6

1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8

1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5

1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6

1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8

1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5

1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6

1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

28.5.8 TIM2 capture/compare mode register 1 [alternate] (TIM2_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
refer to OC1M description on bits 6:4

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2
10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1
11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable
0: tim_oc1ref is not affected by the tim_ocref_clr_int input
1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - In up-counting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In down-counting, channel 1 is inactive (tim_oc1ref = 0) as long as TIMx_CNT>TIMx_CCR1 else active (tim_oc1ref = 1).

0111: PWM mode 2 - In up-counting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In down-counting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved.

1011: Reserved.

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to three clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output.
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

28.5.9 TIM2 capture/compare mode register 2 (TIM2_CCMR2)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

28.5.10 TIM2 capture/compare mode register 2 [alternate] (TIM2_CCMR2)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC3M[3:0]

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4

10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3

11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE**: Output compare 3 clear enable

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR3 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0010: Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0011: Toggle - tim_oc3ref toggles when TIMx_CNT = TIMx_CCR3.

0100: Force inactive level - tim_oc3ref is forced low.

0101: Force active level - tim_oc3ref is forced high.

0110: PWM mode 1 - In up-counting, channel 3 is active as long as TIMx_CNT<TIMx_CCR3 else inactive. In down-counting, channel 3 is inactive (tim_oc3ref = 0) as long as TIMx_CNT>TIMx_CCR3 else active (tim_oc3ref = 1).

0111: PWM mode 2 - In up-counting, channel 3 is inactive as long as TIMx_CNT<TIMx_CCR3 else active. In down-counting, channel 3 is active as long as TIMx_CNT>TIMx_CCR3 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Pulse on compare: a pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bitfields programming in TIMx_ECR.

1011: Direction output. The tim_oc3ref signal is overridden by a copy of the DIR bit.

1100: Combined PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc is the logical OR between tim_oc3ref and tim_oc4ref.

1101: Combined PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc is the logical AND between tim_oc3ref and tim_oc4ref.

1110: Asymmetric PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

28.5.11 TIM2 capture/compare enable register (TIM2_CCER)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Bit 15 **CC4NP**: *Capture/Compare 4 output Polarity.*
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: *Capture/Compare 4 output Polarity.*
Refer to CC1P description

Bit 12 **CC4E**: *Capture/Compare 4 output enable.*
refer to CC1E description

Bit 11 **CC3NP**: *Capture/Compare 3 output Polarity.*
Refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: *Capture/Compare 3 output Polarity.*
Refer to CC1P description

Bit 8 **CC3E**: *Capture/Compare 3 output enable.*
Refer to CC1E description

Bit 7 **CC2NP**: *Capture/Compare 2 output Polarity.*
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: *Capture/Compare 2 output Polarity.*
refer to CC1P description

Bit 4 **CC2E**: *Capture/Compare 2 output enable.*
Refer to CC1E description

Bit 3 **CC1NP**: *Capture/Compare 1 output Polarity.*

CC1 channel configured as output: CC1NP must be kept cleared in this case.

CC1 channel configured as input: This bit is used in conjunction with CC1P to define tim_ti1fp1/tim_ti2fp1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: *Capture/Compare 1 output Polarity.*

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0: this configuration is reserved, it must not be used.

Bit 0 **CC1E**: *Capture/Compare 1 output enable.*

0: Capture mode disabled / OC1 is not active

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

Table 265. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The state of the external IO pins connected to the standard tim_ocx channels depends only on the GPIO registers when CCxE = 0.

28.5.12 TIM2 counter (TIM2_CNT)

Address offset: 0x024

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY_CNT[31]**: Value depends on IUFREMAP in TIMx_CR1.
If UIFREMAP = 0
CNT[31]: Most significant bit of counter value
If UIFREMAP = 1
UIFCPY: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register

Bits 30:0 **CNT[30:0]**: Least significant part of counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register holds the non-dithered part in CNT[30:0]. The fractional part is not available.

28.5.13 TIM2 prescaler (TIM2_PSC)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

28.5.14 TIM2 autoreload register (TIM2_ARR)

Address offset: 0x02C

Reset value: 0xFFFF FFFF

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

Bits 31:0 **ARR[31:0]**: Autoreload value

ARR is the value to be loaded in the actual autoreload register.

Refer to the *Section 28.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the autoreload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the autoreload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[31:4]. The ARR[3:0] bitfield contains the dithered part.

28.5.15 TIM2 capture/compare register 1 (TIM2_CCR1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR1[31:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[31:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[31:0]. The CCR1[3:0] bits are reset.

28.5.16 TIM2 capture/compare register 2 (TIM2_CCR2)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR2[31:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[31:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[31:0]. The CCR2[3:0] bits are reset.

28.5.17 TIM2 capture/compare register 3 (TIM2_CCR3)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR3[31:0]**: Capture/compare 3 value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc3 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR3[31:4]. The CCR3[3:0] bitfield contains the dithered part.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR3[31:0]. The CCR3[3:0] bits are reset.

28.5.18 TIM2 capture/compare register 4 (TIM2_CCR4)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR4[31:0]**: Capture/compare 4 value

If channel CC4 is configured as output:

CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[31:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input:

CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[31:0]. The CCR4[3:0] bits are reset.

28.5.19 TIM2 timer encoder control register (TIM2_ECR)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PWPRSC[2:0]			PW[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS[1:0]		FIDX	IBLK[1:0]		IDIR[1:0]		IE
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

$t_{PW} = PW[7:0] \times t_{PWG}$

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[3:0] = 0001, 0010, 0011, 1110, 1111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[3:0] = 1010, 1011, 1100, 1101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 **IBLK[1:0]**: Index blanking

This bit indicates if the Index event is conditioned by the tim_ti3 input

00: Index always active
01: Index disabled hen tim_ti3 input is active, as per CC3P bitfield
10: Index disabled when tim_ti4 input is active, as per CC4P bitfield
11: Reserved

Bits 2:1 **IDIR[1:0]**: Index direction

This bit indicates in which direction the Index event resets the counter.

00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Bit 0 **IE**: Index enable

This bit indicates if the Index event resets the counter.

0: Index disabled
1: Index enabled

28.5.20 TIM2 timer input selection register (TIM2_TISEL)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 28.4.2: TIM2 pins and internal signals* for product specific implementation.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIMx_CH3
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 28.4.2: TIM2 pins and internal signals* for product specific implementation.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[15:0] input
0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 28.4.2: TIM2 pins and internal signals* for product specific implementation.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[15:0] input
0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 28.4.2: TIM2 pins and internal signals* for product specific implementation.

28.5.21 TIM2 alternate function register 1 (TIM2_AF1)

Address offset: 0x060

Reset value: 0x0000 0000

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

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: etr_in source selection
These bits select the etr_in input source.
0000: tim_etr0: TIMx_ETR input
0001: tim_etr1
...
1111: tim_etr15
Refer to *Section 28.4.2: TIM2 pins and internal signals* for product specific implementation.

Bits 13:0 Reserved, must be kept at reset value.

28.5.22 TIM2 alternate function register 2 (TIM2_AF2)

Address offset: 0x064

Reset value: 0x0000 0000

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection
These bits select the ocref_clr input source.
000: tim_ocref_clr0
001: tim_ocref_clr1
...
111: tim_ocref_clr7
Refer to *Section 28.4.2: TIM2 pins and internal signals* for product specific implementation.

Bits 15:0 Reserved, must be kept at reset value.

28.5.23 TIM2 DMA control register (TIM2_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved
0001: Update
0010: CC1
0011: CC2
0100: CC3
0101: CC4
0110: COM
0111: Trigger
Others: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
11010: 26 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

–If DBL = 7 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer is given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

28.5.24 TIM2 DMA address for full transfer (TIM2_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

28.5.25 TIM2 register map

TIM2 registers are mapped as described in the table below.

Table 266. TIM2 register map and reset values

Offset	Register name	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
0x000	TIM2_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																				0	0		0	0	0	0	0	0	0	0	0	0
0x004	TIM2_CR2	Res.	Res.	Res.	ADSYNC	Res.	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
	Reset value				0			0																		0	0	0	0	0			
0x008	TIM2_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value							0	0			0	0				0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x00C	TIM2_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.	Res.	Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value								0	0	0	0							0		0	0	0	0	0		0		0	0	0	0	0
0x010	TIM2_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value								0	0	0	0									0	0	0	0			0		0	0	0	0	0
0x014	TIM2_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x018	TIM2_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM2_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	TIM2_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM2_CCMR2 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 266. TIM2 register map and reset values (continued)

Offset	Register name	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
0x020	TIM2_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0
0x024	TIM2_CNT	UIFCPY_CNT[31]	CNT[30:16] (CNT[31:16] on 32-bit timers only)															CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	TIM2_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	TIM2_ARR N/A	ARR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x030	Reserved	Reserved																															
0x034	TIM2_CCR1	CCR1[31:20] (32-bit timers only)												CCR1[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	TIM2_CCR2	CCR2[31:20] (32-bit timers only)												CCR2[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	TIM2_CCR3	CCR3[31:20] (32-bit timers only)												CCR3[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	TIM2_CCR4	CCR4[31:20] (32-bit timers only)												CCR4[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044.. 0x054	Reserved	Res.																															
0x058	TIM2_ECR	Res.	Res.	Res.	Res.	Res.	PWPRSC [2:0]			PW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS [1:0]		FIDX	IBLK [1:0]		IDIR [1:0]		IE
	Reset value						0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0
0x05C	TIM2_TISEL	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x060	TIM2_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [3:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0	0	0														
0x064	TIM2_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x068.. 0x3D8	Reserved	Res.																															
0x3DC	TIM2_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0

Table 266. TIM2 register map and reset values (continued)

Offset	Register name	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
0x3E0	TIM2_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

29 General purpose timers (TIM16/TIM17)

29.1 TIM16/TIM17 introduction

The TIM16/TIM17 timers consist of a 16-bit autoreload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM16/TIM17 timers are completely independent, and do not share any resources.

29.2 TIM16/TIM17 main features

The TIM16/TIM17 timers include the following features:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input

29.3 Implementation

Table 267. TIM16/TIM17

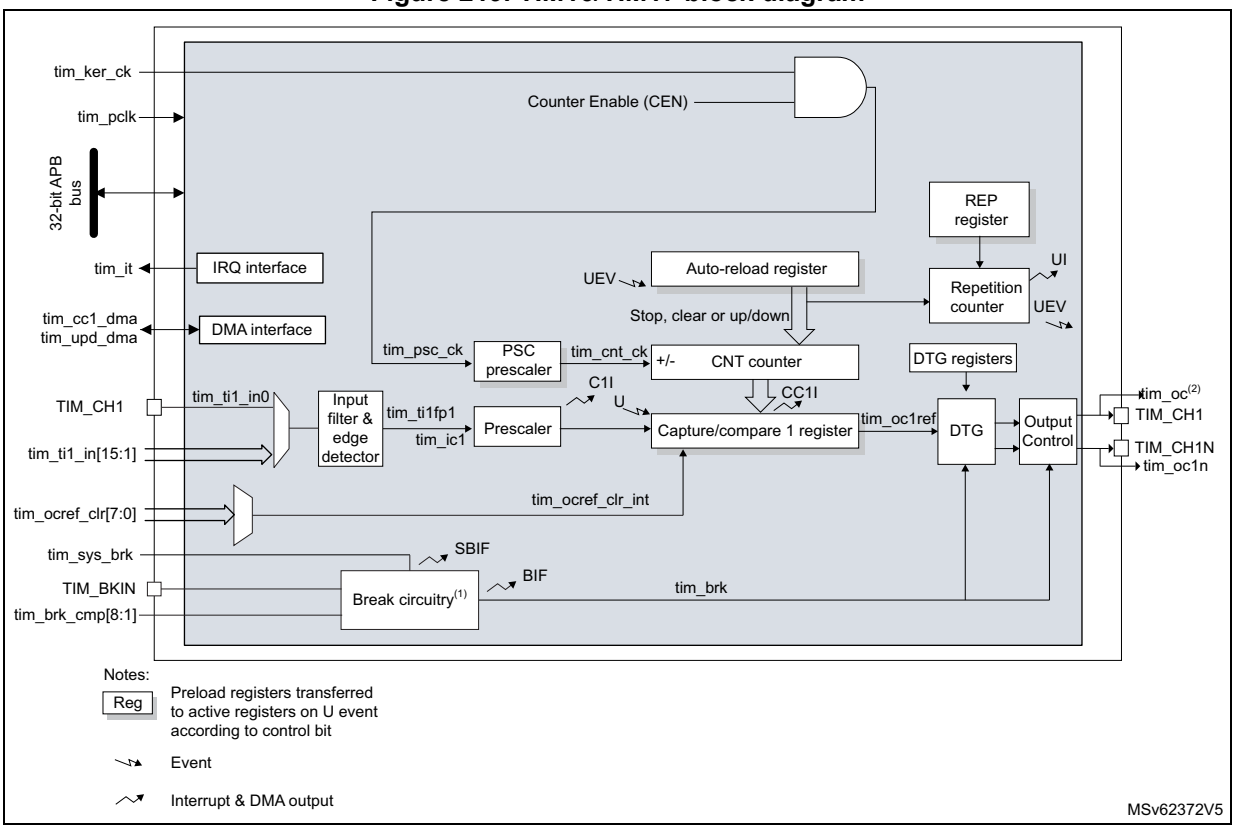
Device	TIM16(1)	TIM17(1)
STM32WBA23	X	X
STM32WBA25	X	X

1. ‘X’ = supported, ‘-’ = not supported.

29.4 TIM16/TIM17 functional description

29.4.1 Block diagram

Figure 219. TIM16/TIM17 block diagram



1. Refer to *Section 29.4.13: Using the break function* for details.
2. This signal can be used as trigger for some slave timer (see internal trigger connection table in next section). See *Section 29.4.19: Using timer output as trigger for other timers (TIM16/TIM17 only)* for details.

29.4.2 TIM16/TIM17 pins and internal signals

Table 268 and *Table 269* in this section summarize the TIM inputs and outputs.

Table 268. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 can also be used as external clock (below 1/4 of the tim_ker_ck clock) and external trigger input.
TIM_CH1N	Output	Timer complementary outputs, derived from TIM_CH1 output with the possibility to have deadtime insertion.
TIM_BKIN	Input / Output	Break input. This input can also be configured in bidirectional mode.

Table 269. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0]	Input	Internal timer input bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock).
tim_itr[15:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[8:1]	Input	Break input for internal signals
tim_sys_brk[n:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_it	Output	Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests
tim_cc1_dma	Output	Timer capture / compare 1 dma request
tim_upd_dma	Output	Timer update dma request

Table 270 lists the sources connected to the tim_ti1 input multiplexer.

Table 270. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources	
	TIM16	TIM17
tim_ti1_in0	TIM16_CH1	TIM17_CH1
tim_ti1_in1	Reserved	
tim_ti1_in2	MCO	
tim_ti1_in3	HSE32 / 32	
tim_ti1_in4	rtc_wut_trg	
tim_ti1_in5	LSE	
tim_ti1_in6	LSI	
tim_ti1_in[8:7]	Reserved	
tim_ti1_in9	HSI16 / 256	
tim_ti1_in[15:10]	Reserved	

Table 271 and *Table 272* list the sources connected to the tim_brk and tim_brk2 inputs.

Table 271. Timer break interconnect

tim_brk inputs	TIM16	TIM17
TIM_BKIN	TIM16_BKIN	TIM17_BKIN
tim_brk_cmp[8:1]	Reserved	

Table 272. System break interconnect

tim_sys_brk inputs	TIM16/TIM17	Enable bit in SYSCFG_CFGR2 register
tim_sys_brk0	Cortex-M33 LOCKUP	CLL
tim_sys_brk1	Programmable voltage detector (PVD)	PVDL
tim_sys_brk2	SRAM parity error	SPL
tim_sys_brk3	Flash ECC error	ECCL
tim_sys_brk4	HSE32 clock security system (HSECSS)	None (always enabled)

Table 273 lists the internal sources connected to the tim_ocref_clr input multiplexer.

Table 273. Interconnect to the ocref_clr input multiplexer

Timer OCREF clear signal	Timer OCREF clear signals assignment	
	TIM16	TIM17
tim_ocref_clr[7:0]	Reserved	

29.4.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting one clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Table 230 and *Table 221* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 220. Counter timing diagram with prescaler division change from 1 to 2

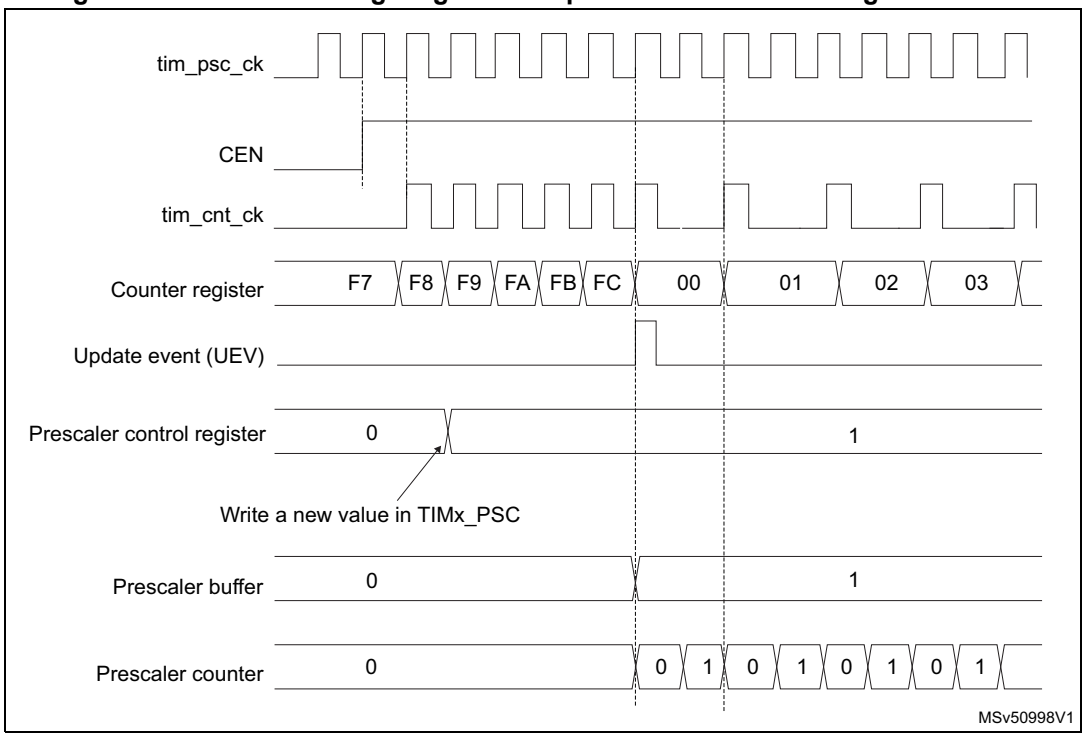
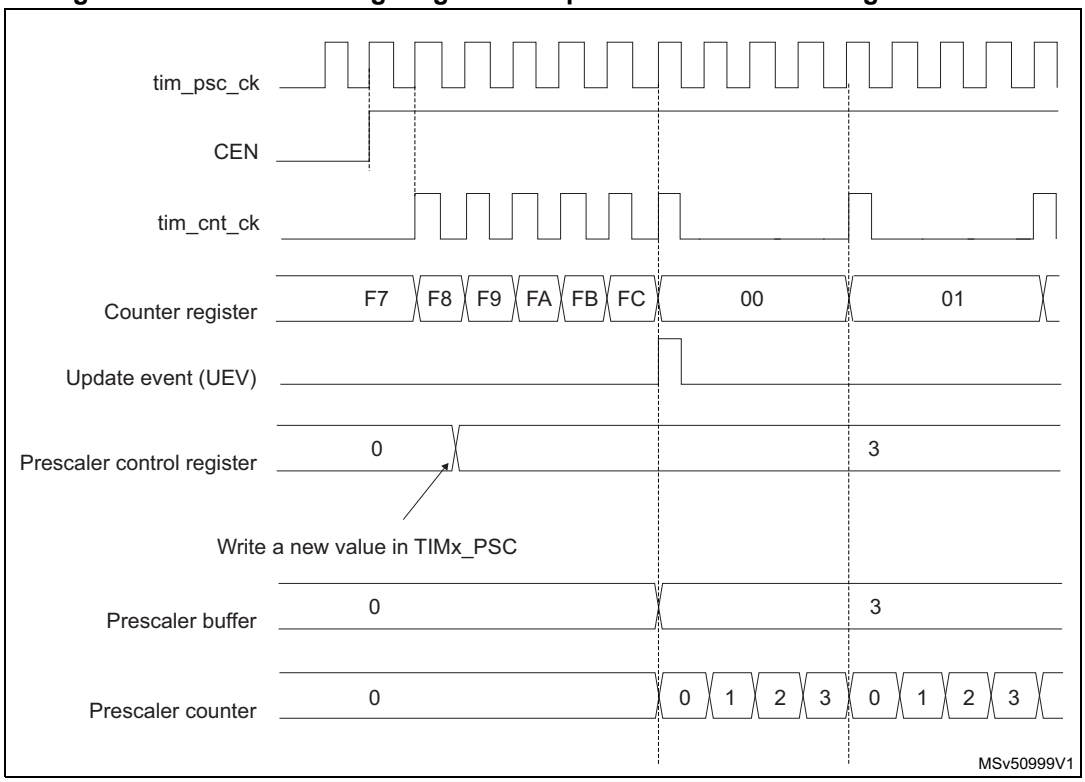


Figure 221. Counter timing diagram with prescaler division change from 1 to 4



29.4.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The autoreload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 222. Counter timing diagram, internal clock divided by 1

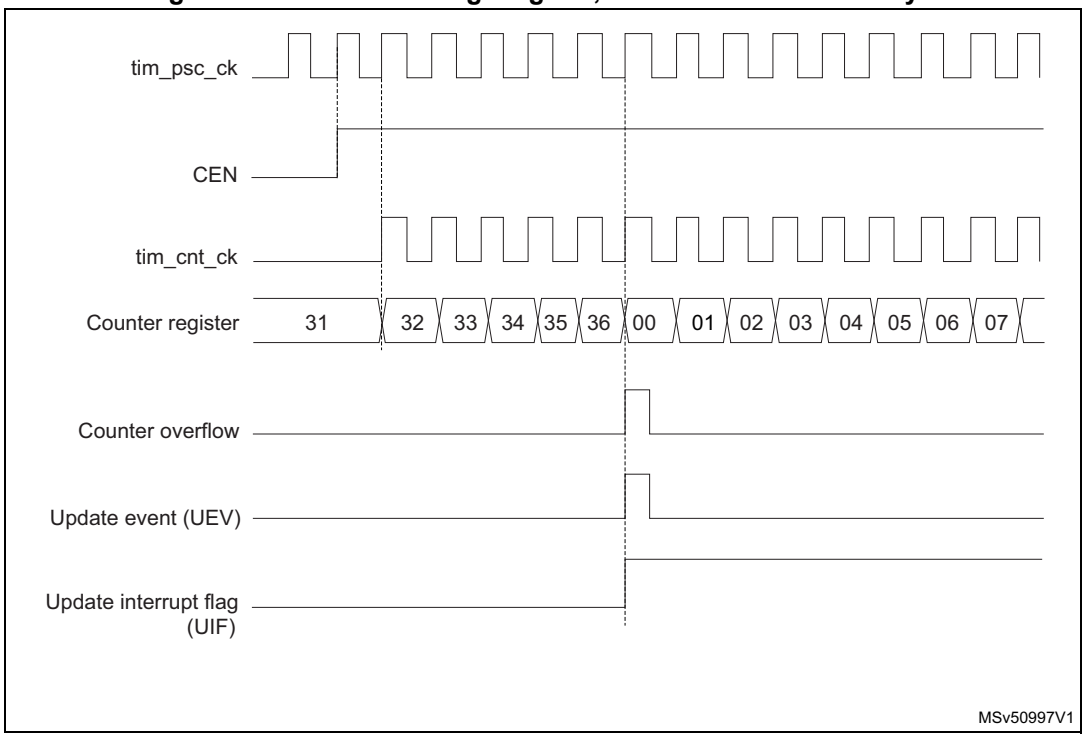


Figure 223. Counter timing diagram, internal clock divided by 2

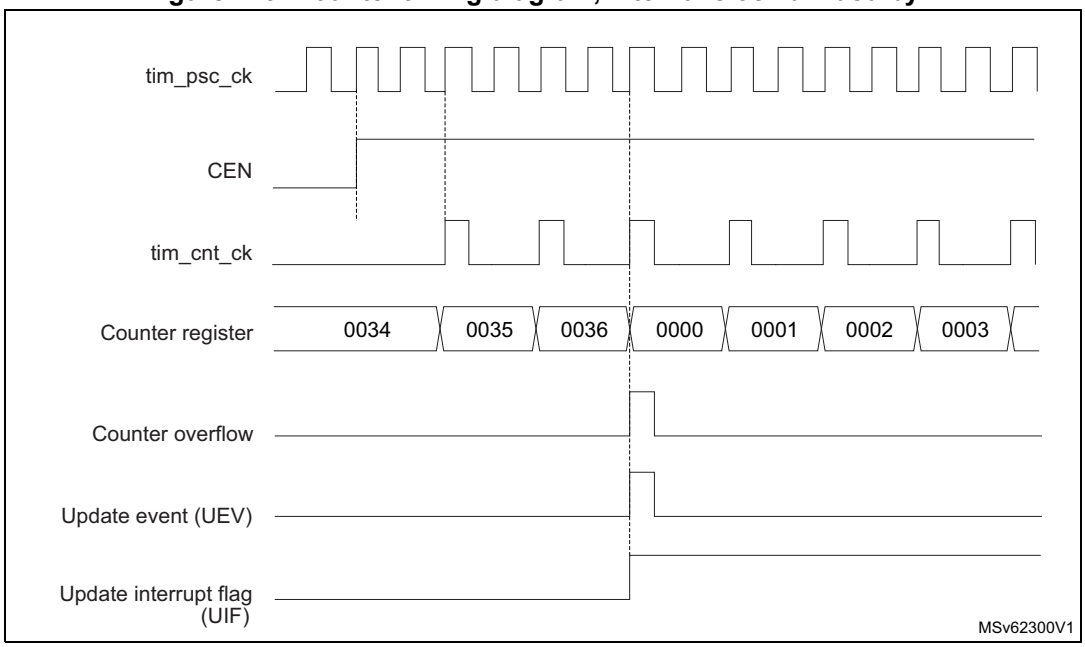


Figure 224. Counter timing diagram, internal clock divided by 4

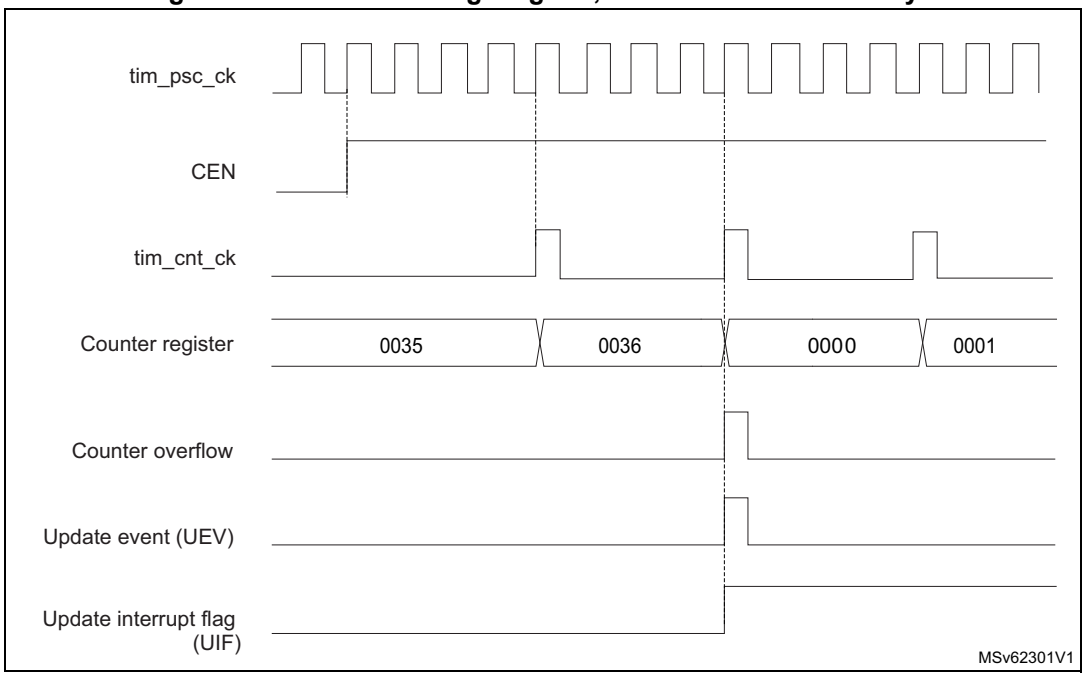


Figure 225. Counter timing diagram, internal clock divided by N

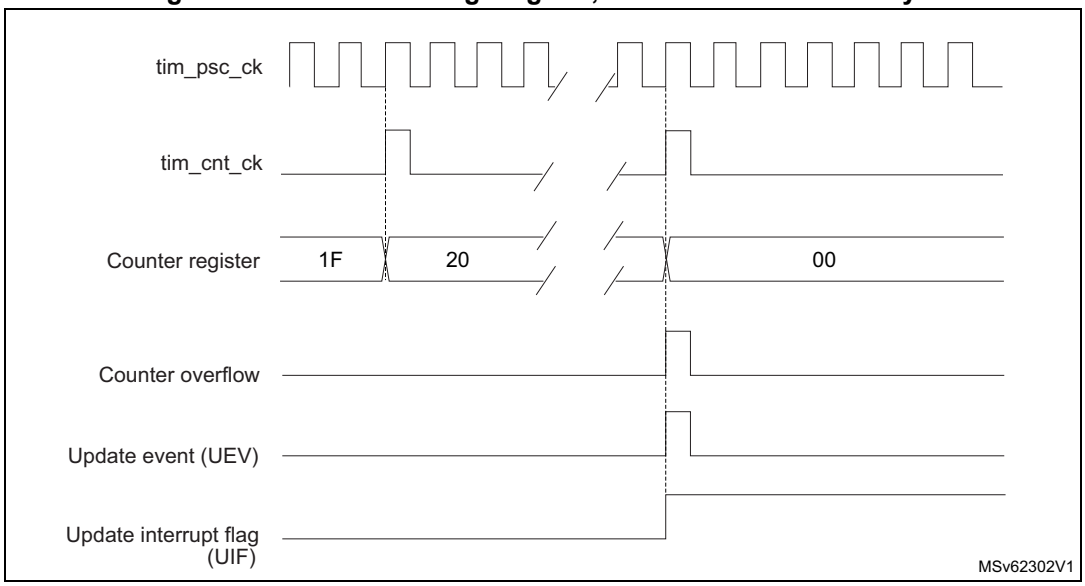


Figure 226. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

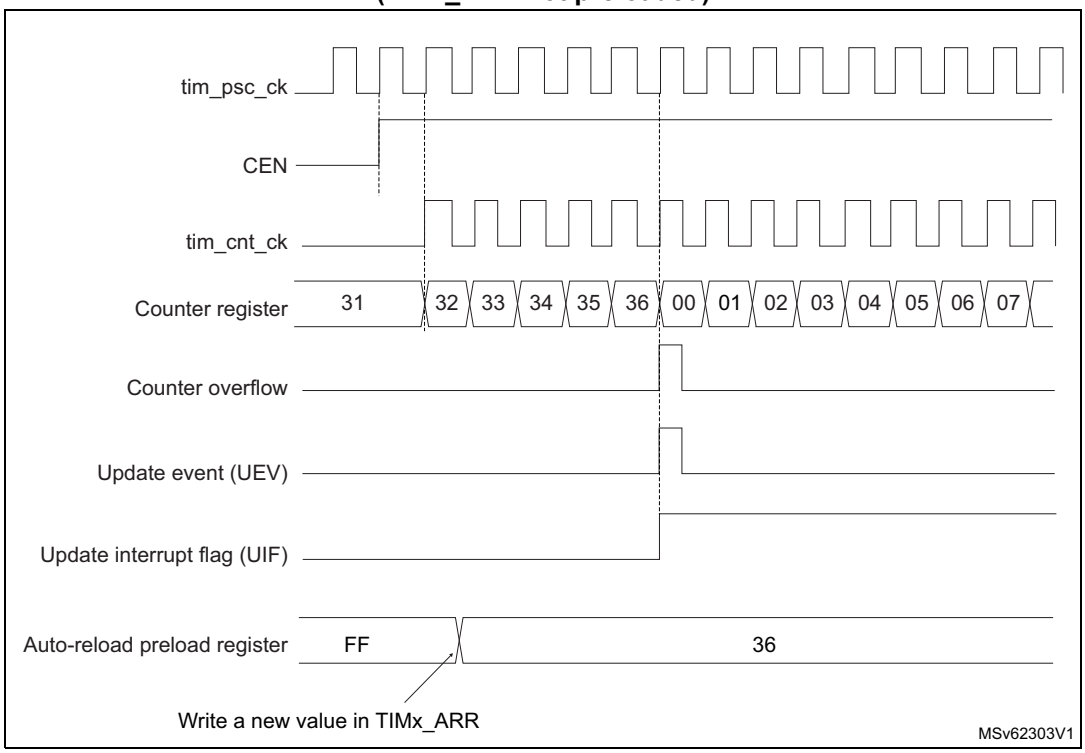
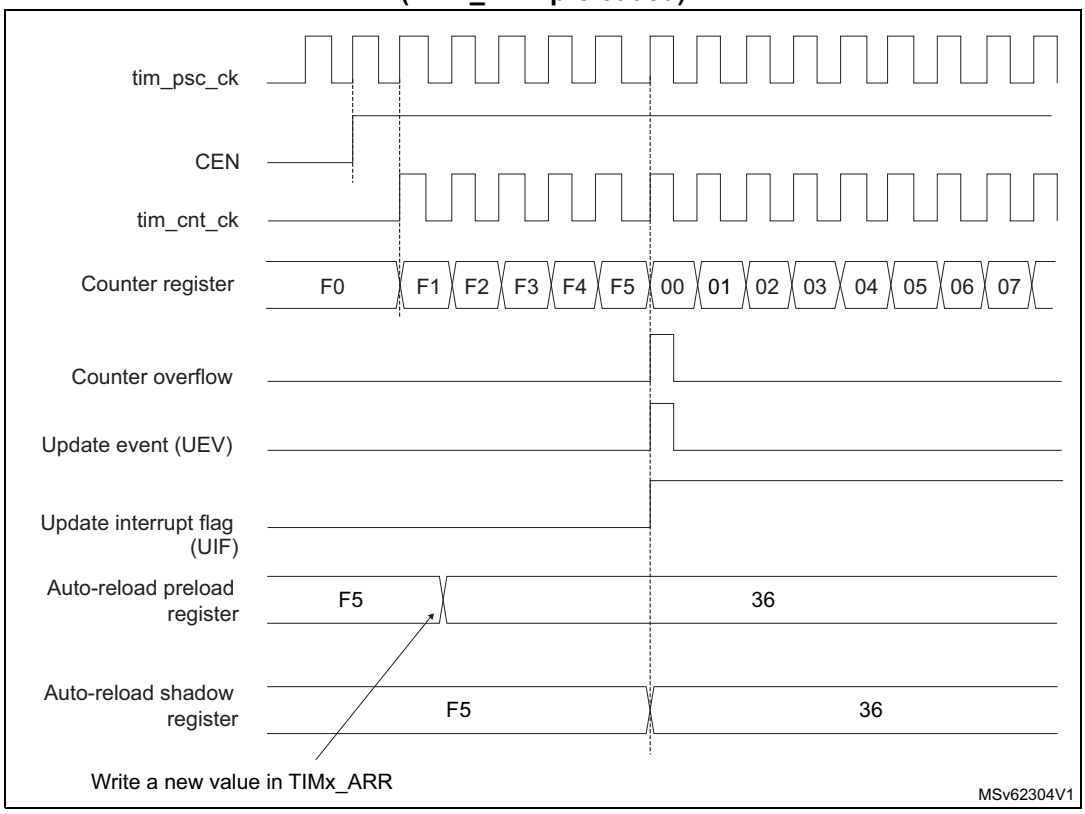


Figure 227. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



29.4.5 Repetition counter

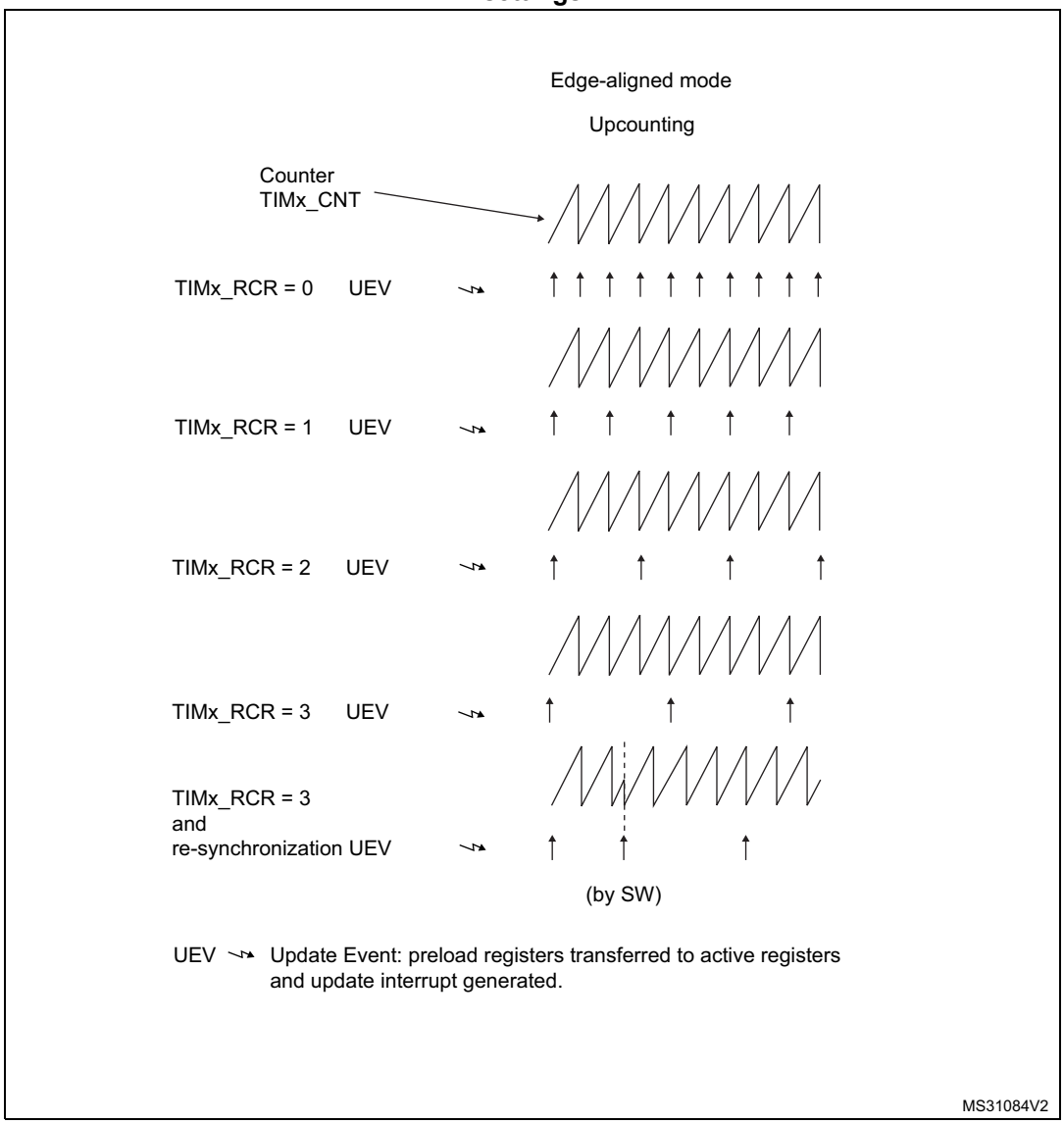
Section 29.4.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR autoreload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an autoreload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 228*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

Figure 228. Update rate examples depending on mode and TIMx_RCR register settings



29.4.6 Clock selection

The counter clock can be provided by the following clock sources:

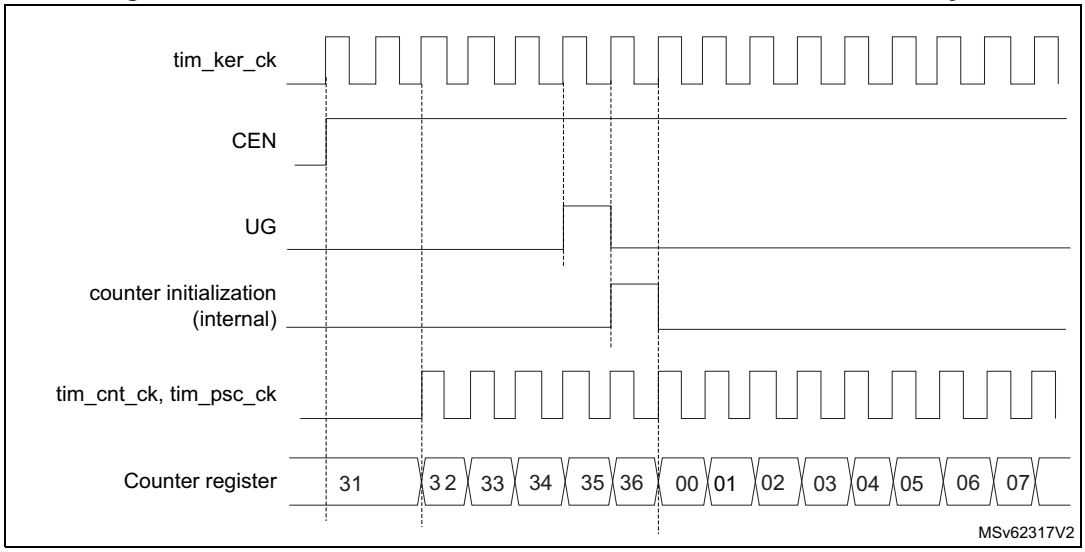
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2, if available)

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000), then the CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 229 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

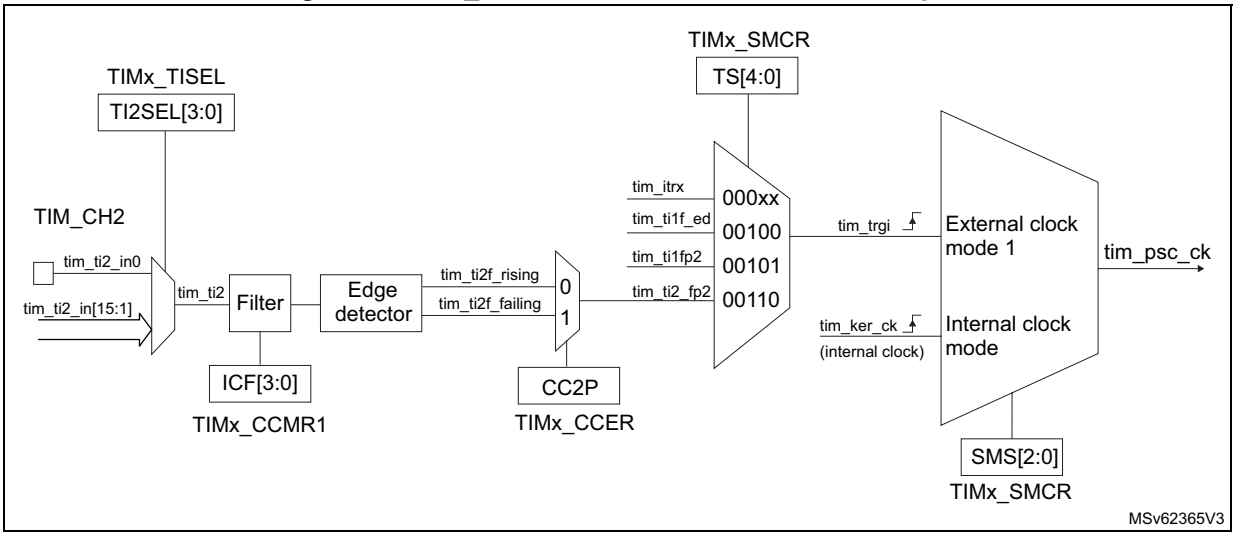
Figure 229. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 230. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

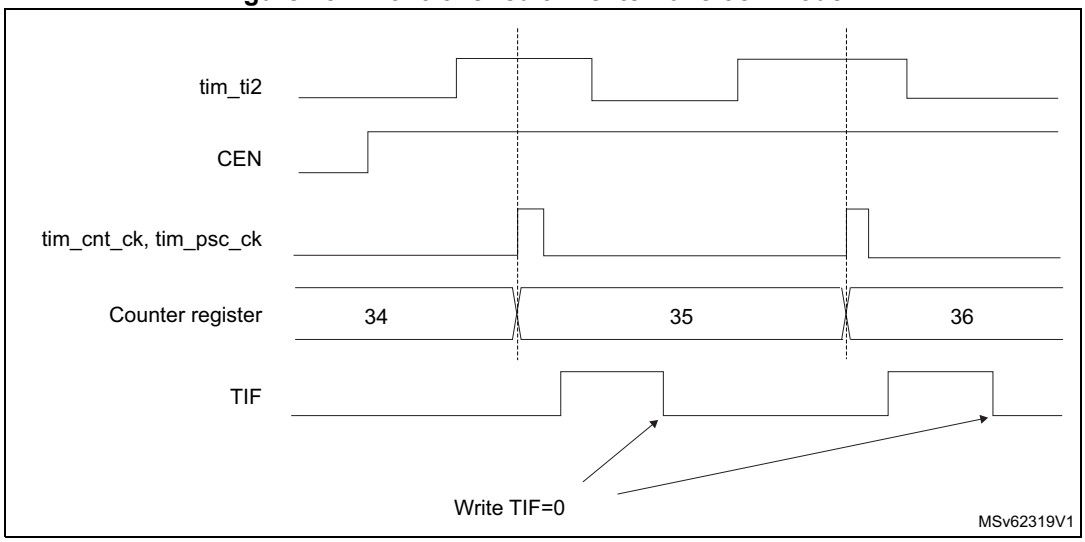
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).
4. Select rising edge polarity by writing CC2P = 0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
6. Select tim_ti2 as the trigger input source by writing TS = 00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 231. Control circuit in external clock mode 1



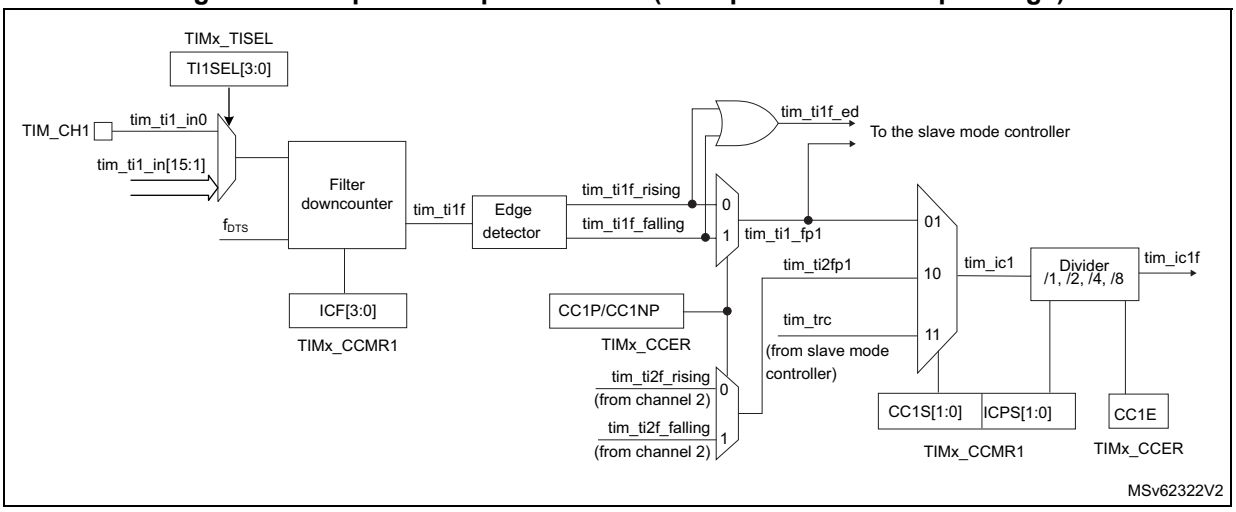
29.4.7 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 232 to *Figure 234* give an overview of one Capture/Compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 232. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 233. Capture/compare channel 1 main circuit

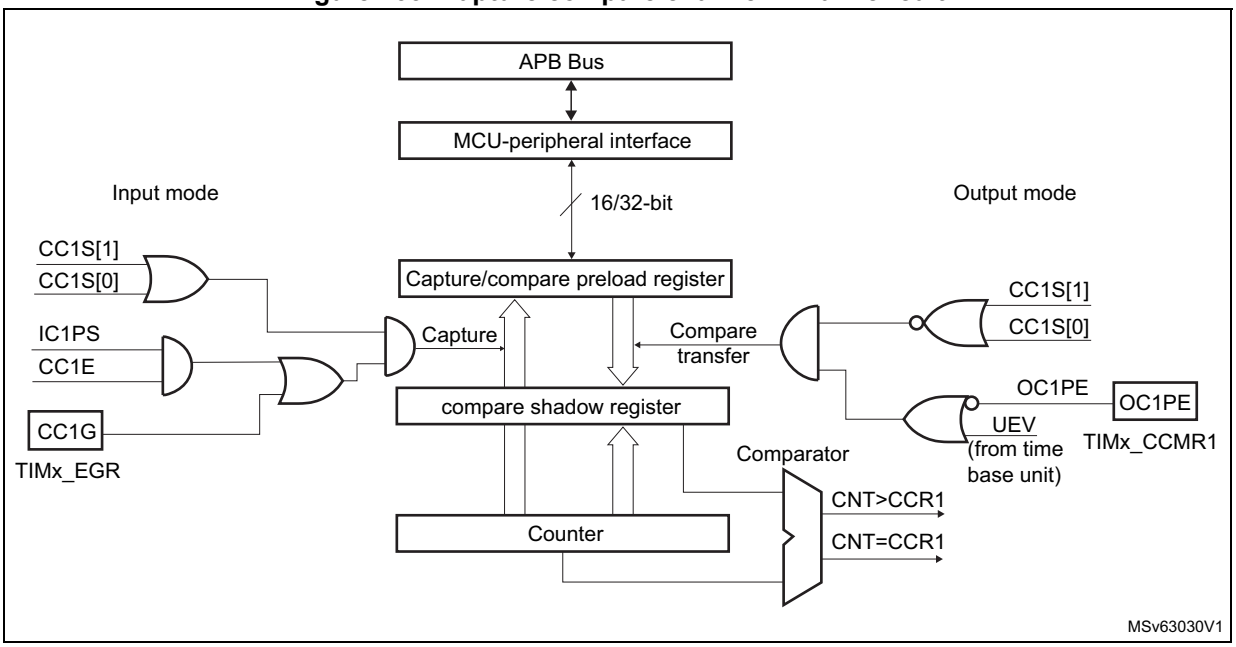
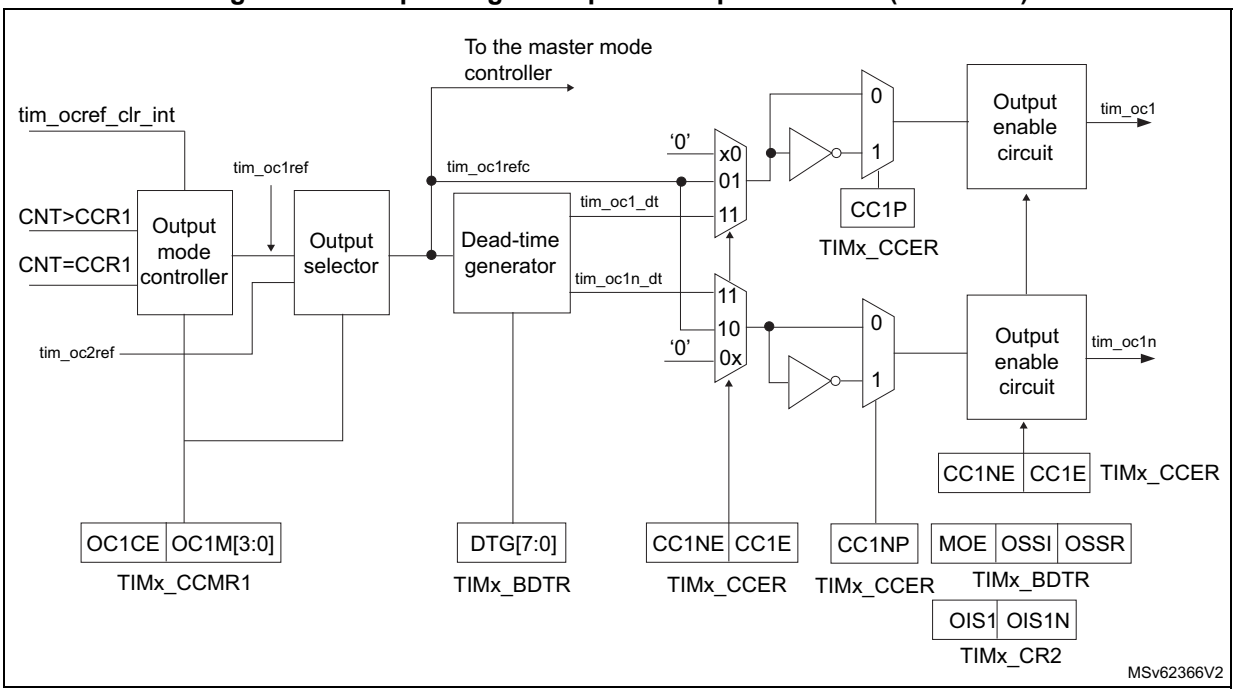


Figure 234. Output stage of capture/compare channel (channel 1)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

29.4.8 Input capture mode

In Input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding tim_icx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_ti1_in[15:1] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input, and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at least 5 internal clock

cycles. The user must program a filter duration longer than these five clock cycles. The user can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

4. Select the edge of the active transition on the tim_ti1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the user wants the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

29.4.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) → tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

29.4.10 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM = 000), be set active (OCxM = 001), be set inactive (OCxM = 010) or can toggle (OCxM = 011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

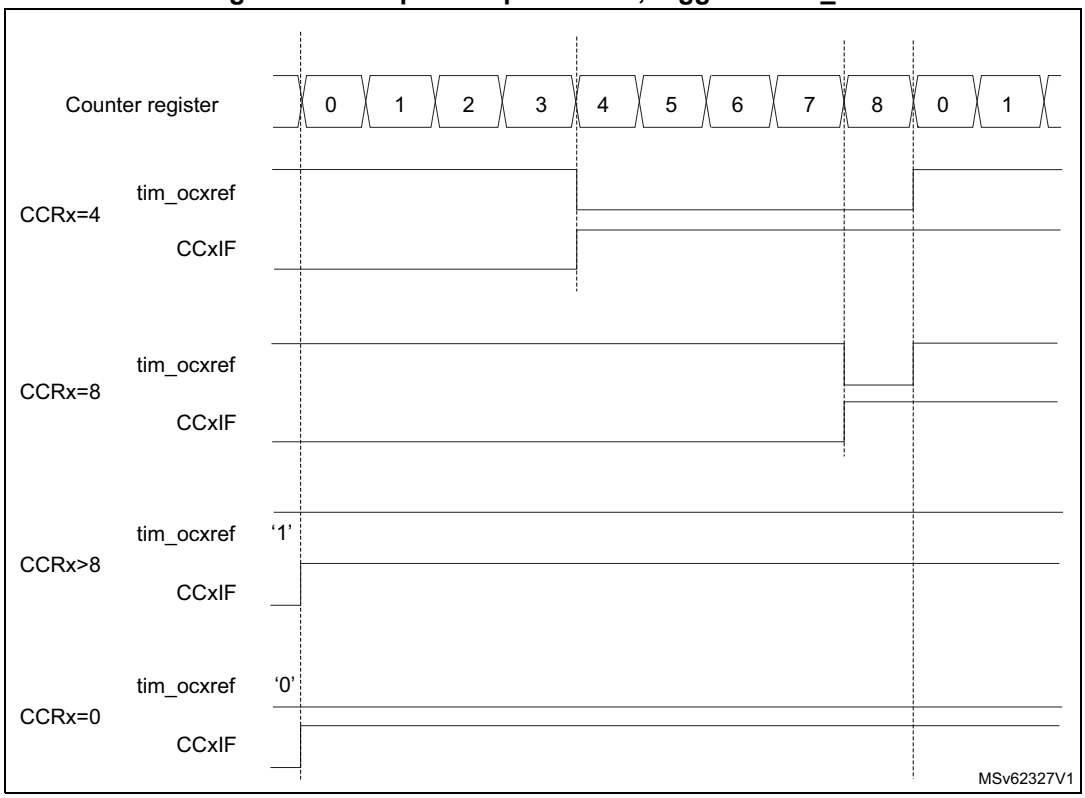
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 235*.

Figure 235. Output compare mode, toggle on tim_oc1



29.4.11 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 110 (PWM mode 1) or 111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI, and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

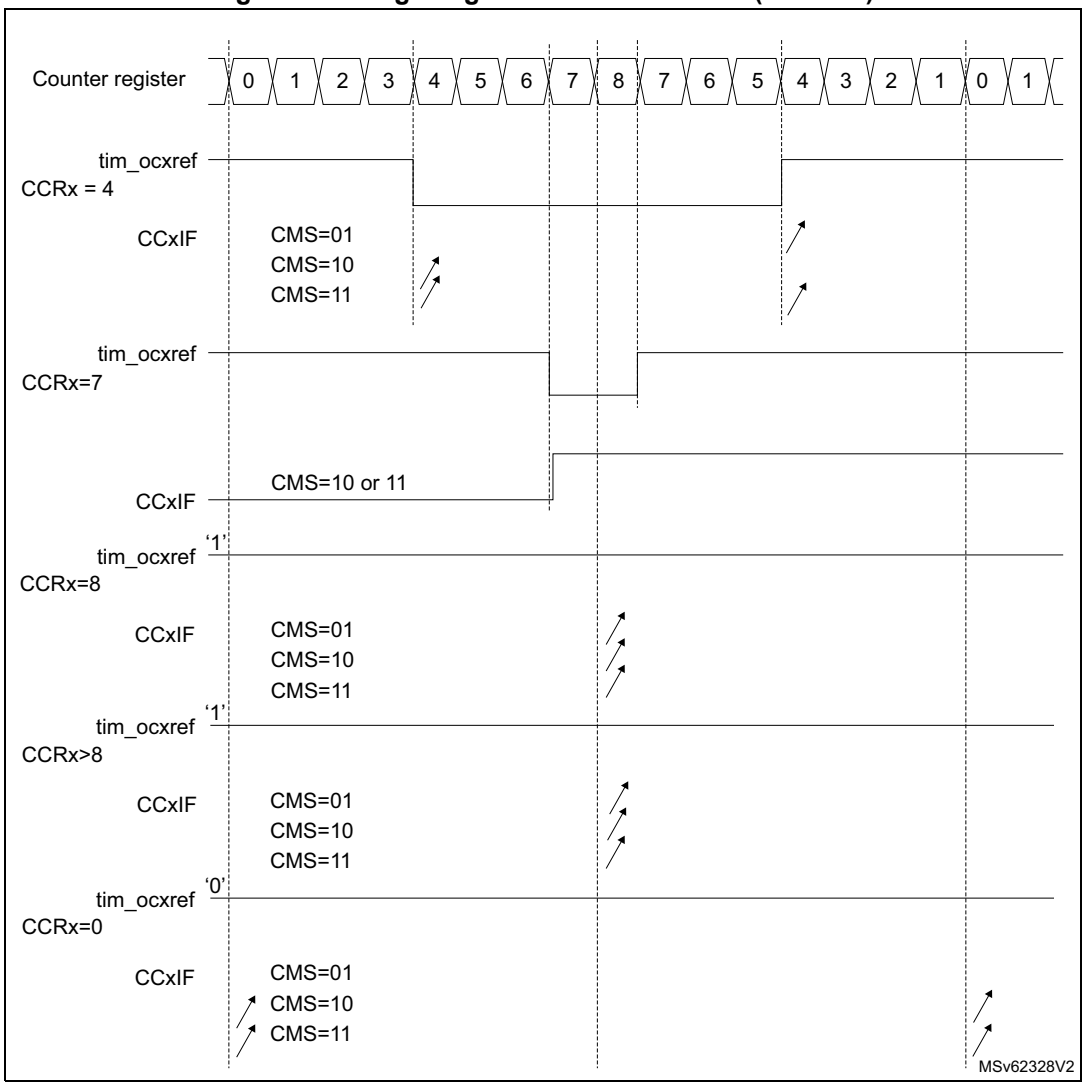
In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The TIM16/TIM17 are capable of upcounting only. Refer to *Upcounting mode*.

In the following example applies to PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in

TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. *Figure 236* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 236. Edge-aligned PWM waveforms (ARR = 8)

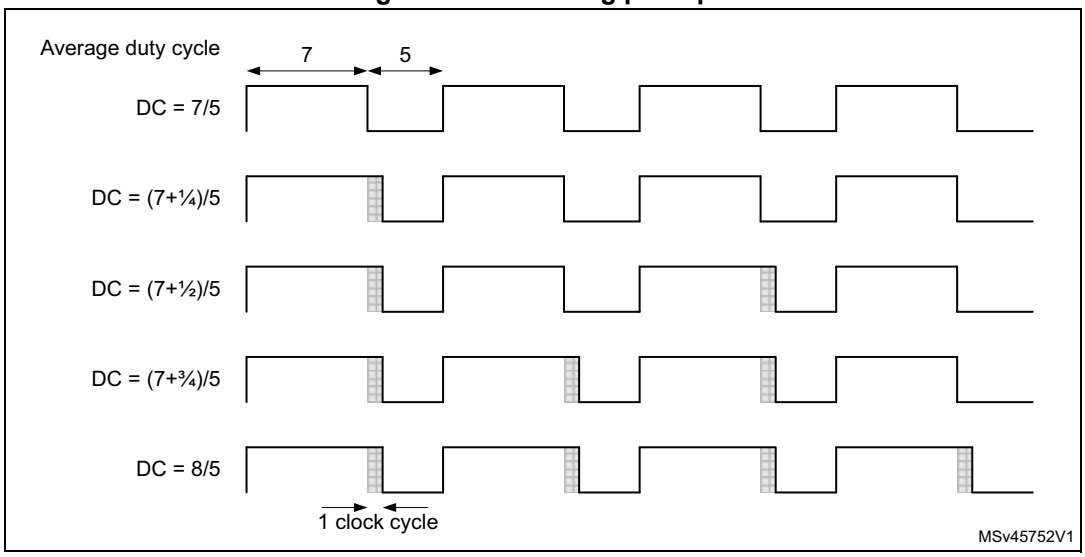


Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. The *Figure 237* below presents the dithering principle applied to four consecutive PWM cycles.

Figure 237. Dithering principle

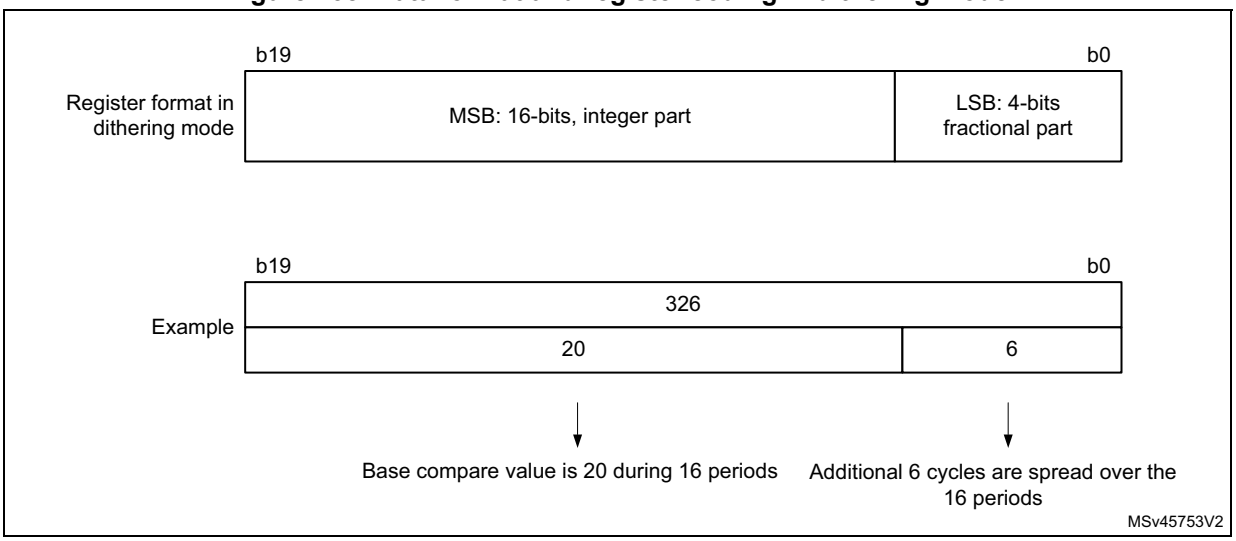


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 238* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The ARR[3:0] bits must be reset
3. The CCIF flags must be cleared
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 238. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

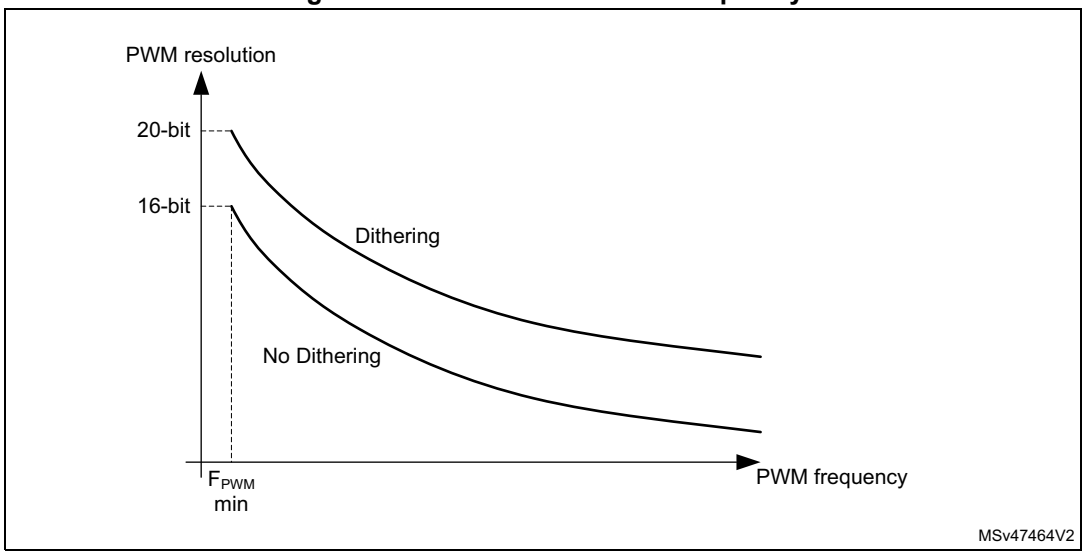
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR and TIMx_CCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

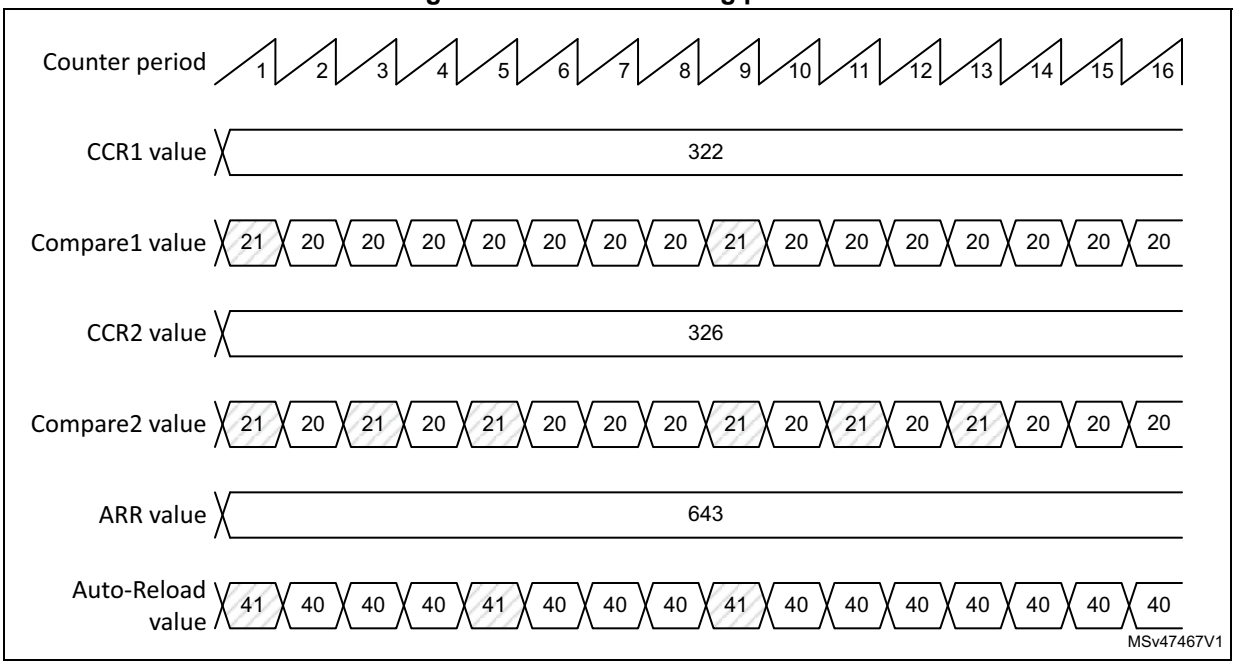
As shown on the *Figure 239* below, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 239. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in the *Figure 240* below.

Figure 240. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in the *Table 274* below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 274. CCR and ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-

Table 274. CCR and ARR register change dithering pattern (continued)

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

29.4.12 Complementary outputs and dead-time insertion

The TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (such as intrinsic delays of level-shifters and delays due to power switches)

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 279: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

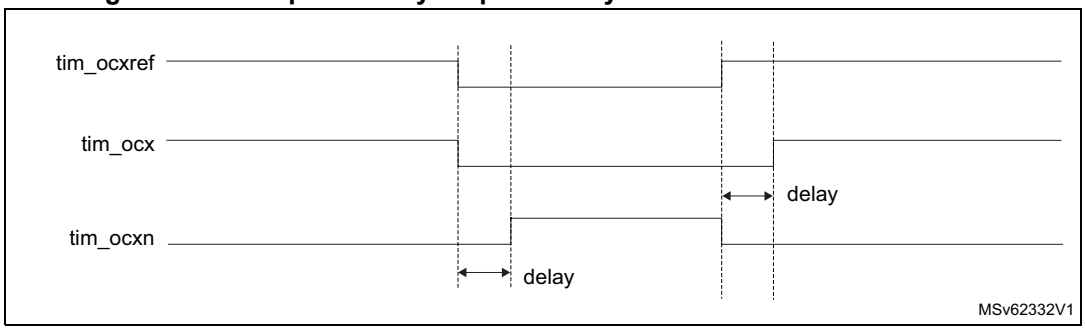
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref. (in these examples CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1)

Figure 241. Complementary output with symmetrical dead-time insertion.



The DTAE bit in the TIMx_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on *Figure 242*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIMx_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIMx_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set in the TIMX_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 242. Asymmetrical deadtime

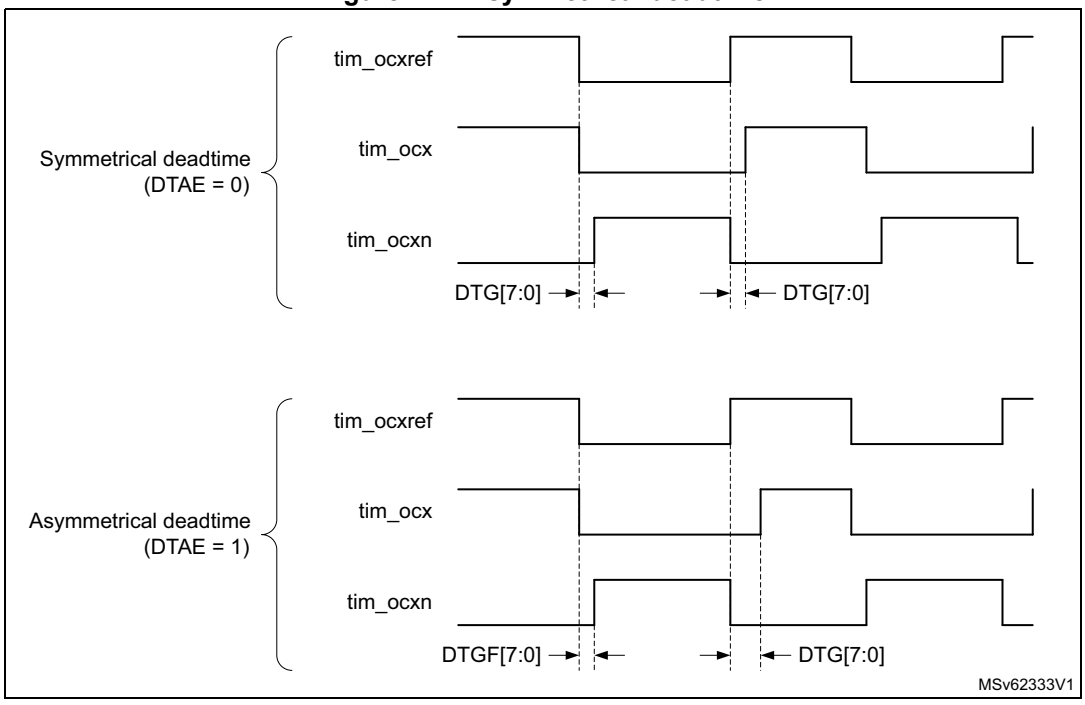


Figure 243. Dead-time waveforms with delay greater than the negative pulse.

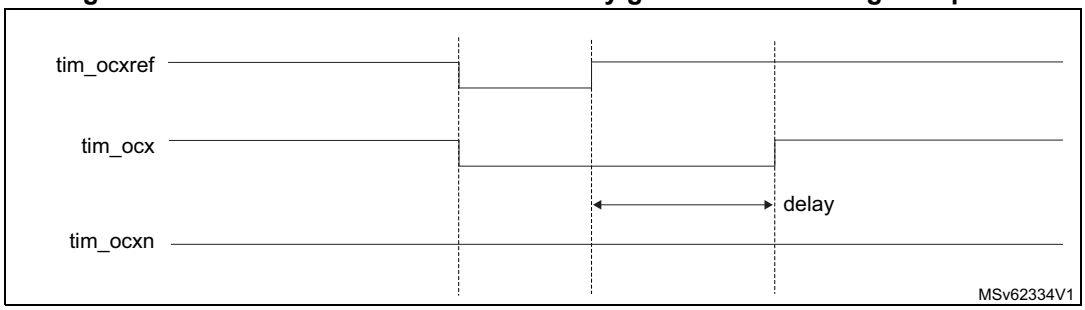
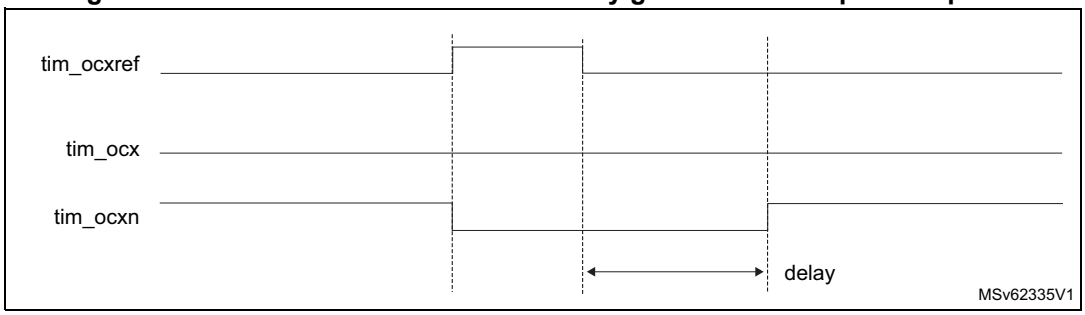


Figure 244. Dead-time waveforms with delay greater than the positive pulse.



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to *Section 29.7.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)* for delay calculation.

Redirecting tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be redirected to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only tim_ocxn is enabled (CCxE = 0, CCxNE = 1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP = 0 then tim_ocxn = tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE = CCxNE = 1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

29.4.13 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (such as clock failure, ECC/parity, and errors) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.

The output enable signal and output levels during break are depending on several control bits:

- The MOE bit in TIMx_BDTR register is used to enable /disable the outputs by software and is reset in case of break or break2 event.
- The OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode).
- The OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 279: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of one APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait one APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The break is generated by the tim_brk inputs which have:

- Programmable polarity (BKP bit in the TIMx_BDTR register).
- Programmable enable bit (BKE bit in the TIMx_BDTR register).
- Programmable filter (BKF[3:0] bits in the TIMx_BDTR register) to avoid spurious events.

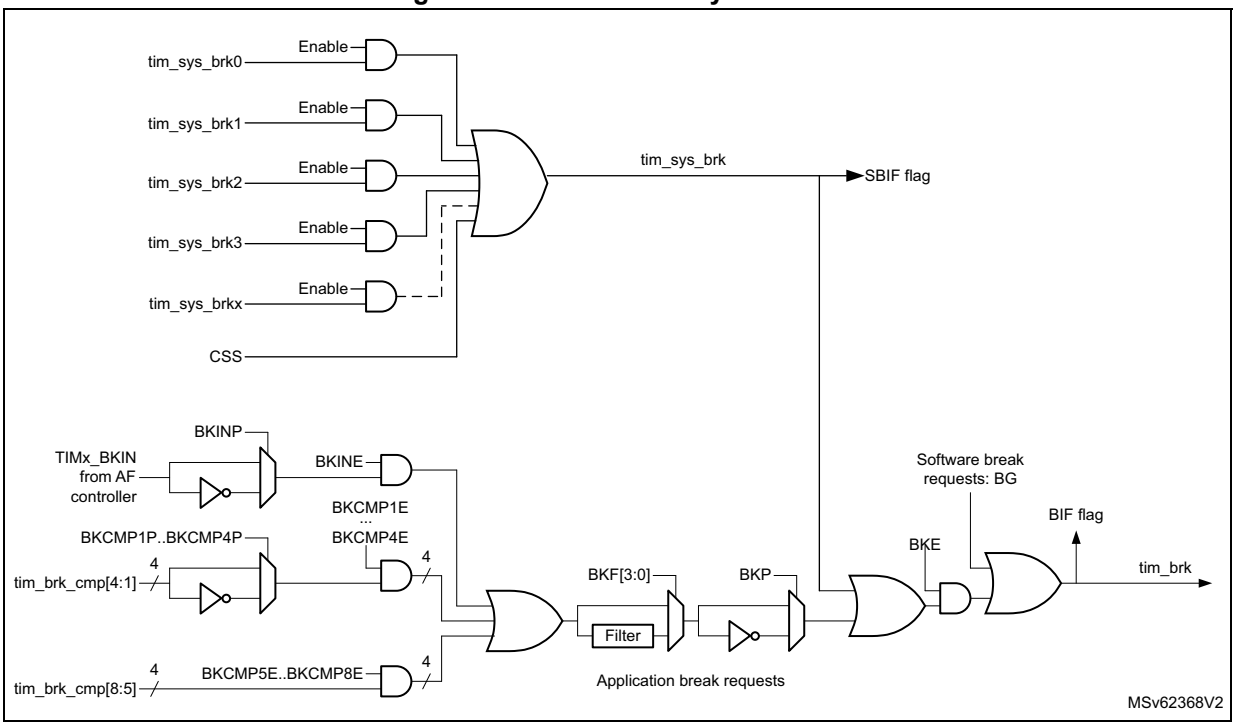
The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_AF1 register.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIM_BKIN pins (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering.
- Internal sources:
  - Coming from a tim_brk_cmpx input (refer to *Section 29.4.2: TIM16/TIM17 pins and internal signals* for product specific implementation).
  - Coming from a system break request on the tim_sys_brk inputs (refer to *Section 29.4.2: TIM16/TIM17 pins and internal signals* for product specific implementation).

Break events can also be generated by software using BG bit in the TIMx_EGR register. All sources are ORed before entering the timer tim_brk inputs, as per *Figure 245* below.

Figure 245. Break circuitry overview



Caution: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail-safe clock mode (for example, using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state, or even releasing the control to the GPIO (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE = 0. If OSSI = 0, the timer releases the output control (taken over by the GPIO) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to their active level together. Note that because of the resynchronization on MOE,

the dead-time duration is a bit longer than usual (around 2 tim_ker_ck clock cycles).

– If OSSI = 0 then the timer releases the enable outputs (taken over by the GPIO which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

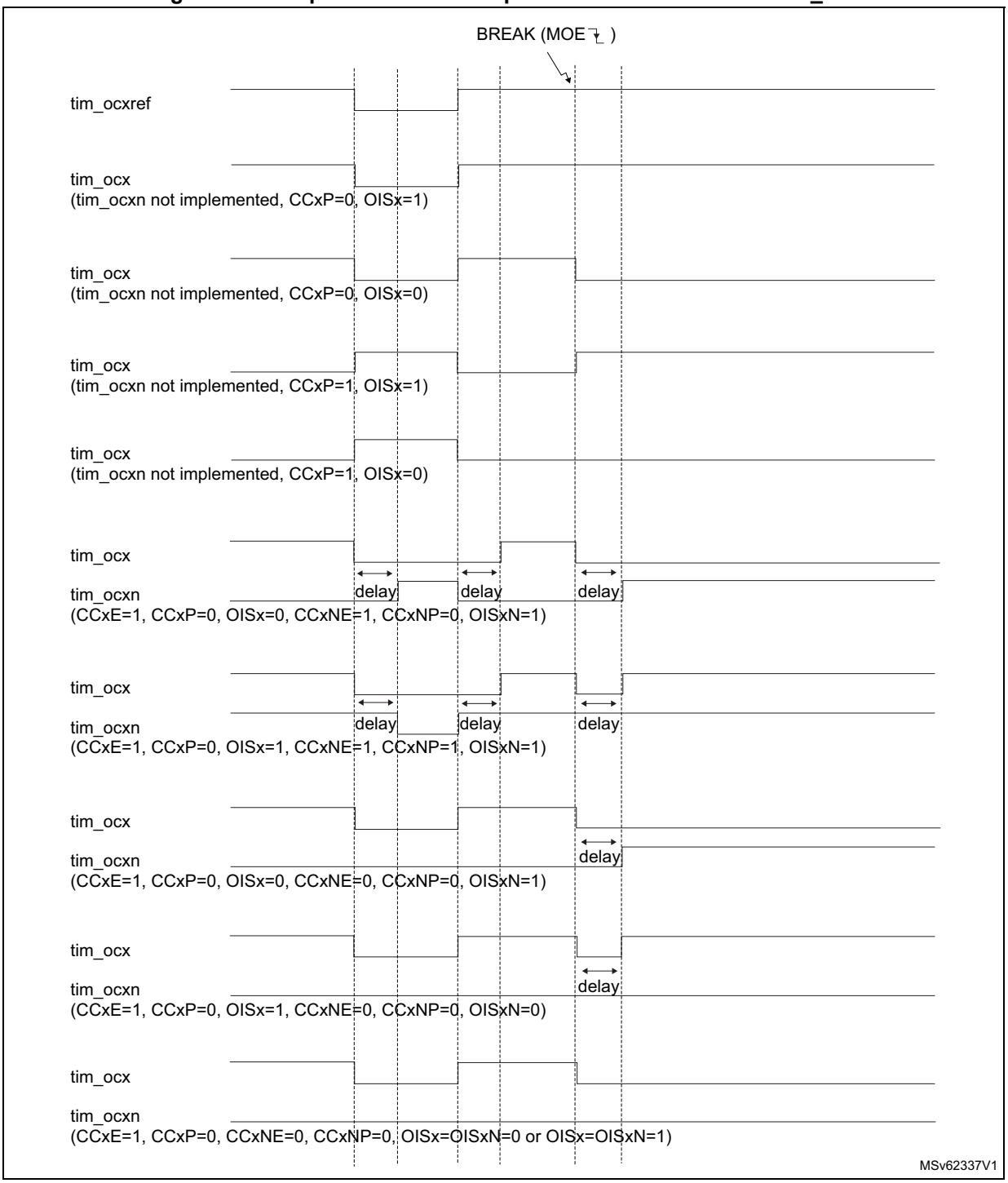
The break inputs are acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the tim_brk input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable, and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIMx_BDTR register. Refer to *Section 29.7.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)*. The LOCK bits can be written only once after an MCU reset.

The *Figure 246* shows an example of behavior for the outputs in response to a break.

Figure 246. Output behavior in response to a break event on tim_brk



29.4.14 Bidirectional break input

The TIM16/TIM17 are featuring bidirectional break I/Os, as represented on *Figure 247*.

They are used to have:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin.
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.

The tim_brk input is configured in bidirectional mode using the BKBID bit in the TIMx_BDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIMx_BDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming either from system (for example CSS), from on-chip peripherals, or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (triggered by setting the BG bit) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the TIM_BKIN I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output to be restarted as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 275*).

Table 275. Break protection disarming conditions

MOE	BKBID	BKDSRM	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and rearming break circuitry

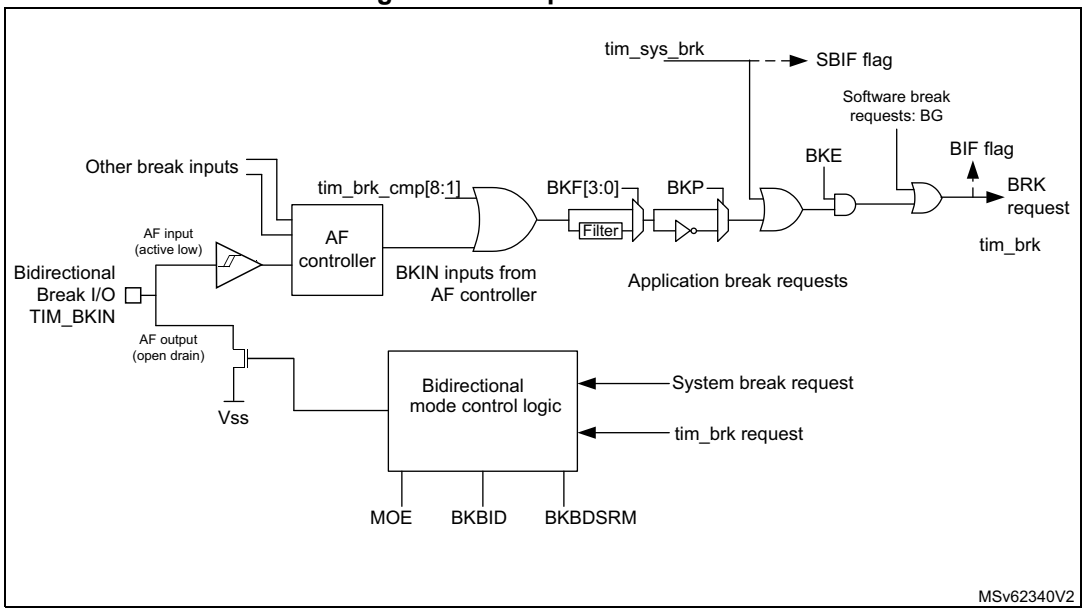
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control.
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before rearming).
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears).

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 247. Output redirection

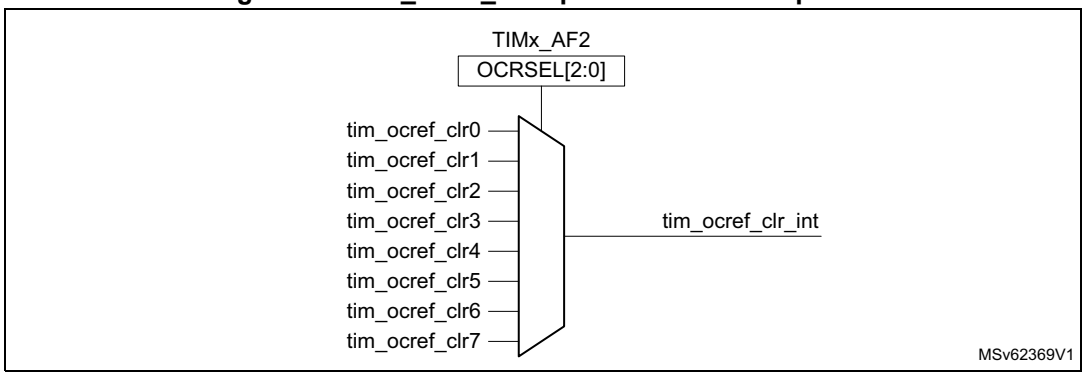


29.4.15 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The tim_ocref_clr_int input can be selected among several inputs, as shown on *Figure 509* below.

Figure 248. tim_ocref_clr input selection multiplexer



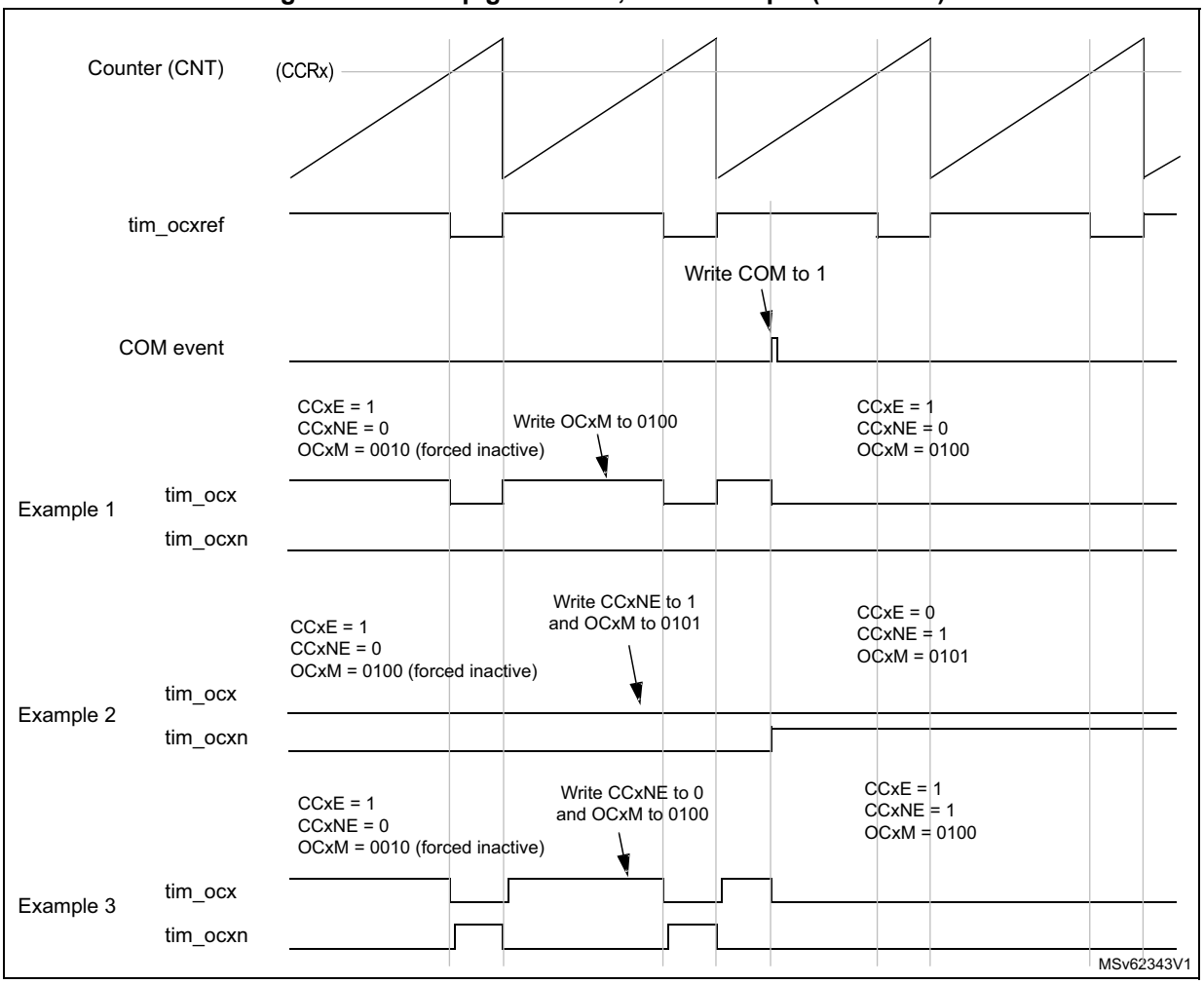
29.4.16 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE, and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The *Figure 249* describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 249. 6-step generation, COM example (OSSR = 1)



29.4.17 One-pulse mode

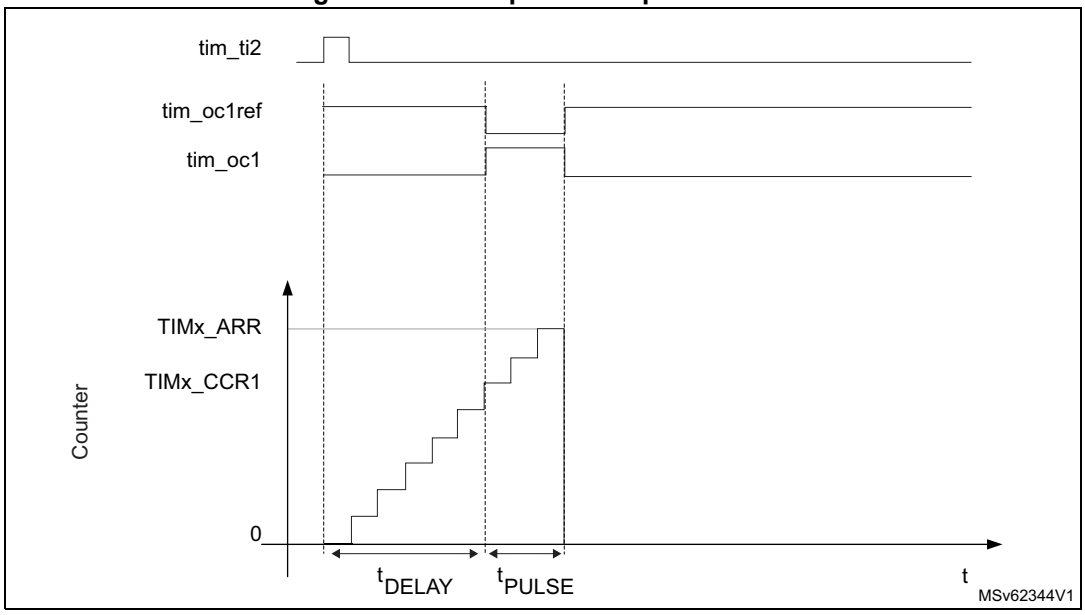
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- CNT < CCRx ≤ ARR (in particular, 0 < CCRx).

Figure 250. Example of one pulse mode.



For example one may want to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Let's use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:1] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

Since only one pulse is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0).

Particular case: tim_ocx fast enable

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be obtained.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

29.4.18 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into bit 31 of the timer counter register (TIMx_CNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

29.4.19 Using timer output as trigger for other timers (TIM16/TIM17 only)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the "TIMx internal trigger connection" table of any timer on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK_INT clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

29.4.20 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to reprogram several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write accesses are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 29.7.20: TIMx DMA control register (TIMx_DCR)(x = 16 to 17)* for details).

For example, the timer DMA burst feature can be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

29.4.21 TIM16/TIM17 DMA requests

The TIM16/TIM17 can generate a DMA request, as shown in *Table 276*.

Table 276. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE

29.4.22 Debug mode

When the microcontroller enters debug mode (Core core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

For more details, refer to the Debug section.

29.5 TIM16/TIM17 low-power modes

Table 277. Effect of low-power modes on TIM16/TIM17

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop 0, and 1	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Stop 2, 3, and Standby	The timer is powered-down and must be reinitialized after exiting the Stop 2, 3, and Standby modes.

29.6 TIM16/TIM17 interrupts

The TIM16/TIM17 can generate multiple interrupts, as shown in *Table 278*.

Table 278. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM	Update	UIF	UIE	write 0 in UIF	Yes	No
	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	Trigger	TIF	TIE	write 0 in TIF	Yes	No
	Break	BIF	BIE	write 0 in BIF	Yes	No

29.7 TIM16/TIM17 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

29.7.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_tix),
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: nly counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

29.7.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
						rw	rw					rw	rw		rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)

0: tim_oc1n = 0 after a dead-time when MOE = 0

1: tim_oc1n = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)

0: tim_oc1 = 0 after a dead-time when MOE = 0

1: tim_oc1 = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when a rising edge occurs on tim_trgi (if available).

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.
Note: This bit acts only on channels that have a complementary output.

29.7.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
						rw	rw	rw		rw				rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

29.7.4 TIMx status register (TIMx_SR)(x = 16 to 17)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
						rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the tim_brk input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred

1: An active level has been detected on the break input

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.

0: No COM event occurred

1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred

If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

29.7.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
								w		w				w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

29.7.6 TIMx capture/compare mode register 1 (TIMx_CCMR1) (x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N =
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input.
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

29.7.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OC1CE**: Output Compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr input.

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr input.

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

29.7.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1. Refer to the description of CC1P.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 279* for details.

Table 279. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)

Control bits					Output states[1]	
MOE bit	OSSI bit	OSSR bit	CC1E bit	CC1NE bit	tim_oc1 output state	tim_oc1n output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) tim_oc1 = 0 tim_oc1n = 0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) tim_oc1 = 0	tim_oc1ref + Polarity tim_oc1n = tim_oc1ref XOR CC1NP
		0	1	0	tim_oc1ref + Polarity tim_oc1 = tim_oc1ref XOR CC1P	Output Disabled (not driven by the timer: Hi-Z) tim_oc1n = 0
		X	1	1	tim_oc1ref + Polarity + dead-time	Complementary to tim_oc1ref (not tim_oc1ref) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_oc1 = CC1P	tim_oc1ref + Polarity tim_oc1n = tim_oc1ref XOR CC1NP
		1	1	0	tim_oc1ref + Polarity tim_oc1 = tim_oc1ref XOR CC1P	Off-State (output enabled with inactive state) tim_oc1n = CC1NP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z)	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_oc1 = CC1P, tim_oc1n = CC1NP Then if the clock is present: tim_oc1 = OIS1 and tim_oc1n = OIS1N after a dead-time, assuming that OIS1 and OIS1N do not correspond to tim_oc1 and tim_oc1n both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by the GPIO controller), the OIS1, OIS1N, CC1P and CC1NP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_oc1 and tim_oc1n channels depends on the tim_oc1 and tim_oc1n channel state and GPIO control and alternate function selection registers.

29.7.9 TIMx counter (TIMx_CNT)(x = 16 to 17)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

29.7.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (tim_cnt_ck) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

29.7.11 TIMx auto-reautoreload register (TIMx_ARR)(x = 16 to 17)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 29.4.3: Time-base unit on page 979* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

29.7.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

– The number of PWM periods in edge-aligned mode.
– The number of half PWM period in center-aligned mode.

29.7.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.
If channel CC1 is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1).
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

29.7.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BK DSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_oc1 and tim_oc1n outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_oc1 and tim_oc1n outputs are enabled if their respective enable bits are set (CC1E, CC1NE in TIMx_CCER register)

See tim_oc1/tim_oc1n enable description for more details (*Section 29.7.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the tim_brk input is not active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low
1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk event) disabled
1; Break inputs (tim_brk and tim_sys_brk event) enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_oc1/tim_oc1n enable description for more details (*Section 29.7.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, tim_oc1/tim_oc1n outputs are enabled with their inactive level as soon as CC1E = 1 or CC1NE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE = 0 on channels configured as outputs.

See tim_oc1/tim_oc1n enable description for more details (*Section 29.7.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (tim_oc1/tim_oc1n enable output signal = 0)

1: When inactive, tim_oc1/tim_oc1n outputs are forced first with their idle level as soon as CC1E = 1 or CC1NE = 1. tim_oc1/tim_oc1n enable output signal = 1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKBID/BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx → DT = DTG[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$

DTG[7:5] = 10x → DT = (64+DTG[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$

DTG[7:5] = 110 → DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$

DTG[7:5] = 111 → DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$

Example if T_{DTS}=125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

29.7.15 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 16 to 17)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx → DTF = DTGF[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$.

DTGF[7:5] = 10x → DTF = (64+DTGF[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$.

DTGF[7:5] = 110 → DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$.

DTGF[7:5] = 111 → DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$.

Example if T_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

29.7.16 TIMx input selection register (TIMx_TISEL)(x = 16 to 17)

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input
0000: TIMx_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 29.4.2: TIM16/TIM17 pins and internal signals* for interconnects list.

29.7.17 TIMx alternate function register 1 (TIMx_AF1)(x = 16 to 17)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 29.4.2: TIM16/TIM17 pins and internal signals* for product specific implementation.

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity
This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp4 input is active high
1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

29.7.18 TIMx alternate function register 2 (TIMx_AF2)(x = 16 to 17)

Address offset: 0x064

Reset value: 0x0000 0000

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: tim_ocref_clr source selection

These bits select the tim_ocref_clr input source.
000: tim_ocref_clr0
001: tim_ocref_clr1
010: tim_ocref_clr2
011: tim_ocref_clr3
100: tim_ocref_clr4
101: tim_ocref_clr5
110: tim_ocref_clr6
111: tim_ocref_clr7
Refer to *Section 29.4.2: TIM16/TIM17 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

29.7.19 TIMx option register 1 (TIMx_OR1)(x = 16 to 17)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSE32 EN	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **HSE32EN**: HSE divided by 32 enable

This bit enables the HSE divider by 32 for the tim_ti1_in3. See *Table 270: Interconnect to the tim_ti1 input multiplexer* for details.
0: HSE divided by 32 disabled
1: HSE divided by 32 enabled

Bit 0 Reserved, must be kept at reset value.

29.7.20 TIMx DMA control register (TIMx_DCR)(x = 16 to 17)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved

0001: Update

0010: CC1

Other: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bitfield defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bitfield defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

29.7.21 TIM16/TIM17 DMA address for full transfer (TIMx_DMAR)(x = 16 to 17)

Address offset: 0x3E0

Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

29.7.22 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

Table 280. TIM16/TIM17 register map and reset values

Offset	Register name	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
0x00	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	TIMx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
	Reset value																							0	0					0	0		0
0x08	Reserved	Res.																															
0x0C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE		COMIE	Res.	Res.	Res.	CC1IE	UIE
	Reset value																							0	0	0		0				0	0
0x10	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
	Reset value																							0		0		0				0	0
0x14	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
	Reset value																									0		0				0	0
0x18	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M [2:0]				OC1PE	OC1FE	CC1 S [1:0]	
	Reset value																0									0	0	0	0		0	0	0	0
0x1C	Reserved	Res.																															
0x20	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
	Reset value																													0	0	0	0
0x24	TIMx_CNT	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 280. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	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
0x2C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	TIMx_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	TIMx_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 - 0x40	Reserved	Res.																															
0x44	TIMx_BDTR	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value				0		0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48 - 0x50	Reserved	Res.																															
0x54	TIMx_DTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x58	Reserved	Res.																															
0x5C	TIMx_TISEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0
0x60	TIMx_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	1	
0x64	TIMx_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCR SEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x68	TIMx_OR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSE32EN	Res.	
	Reset value																														0		
0x6C - 0x3D8	Reserved	Res.																															
0x3DC	TIMx_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	TIMx_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

30 Low-power timer (LPTIM)

30.1 LPTIM introduction

The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. The LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

The LPTIM introduces a flexible clock scheme that provides the needed functionalities and performance, while minimizing the power consumption.

30.2 LPTIM main features

- 16-bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: configurable internal clock source (see RCC section)
  - External clock source over LPTIM input (working with no LP oscillator running, used by pulse counter application)
- 16-bit ARR autoreload register
- 16-bit capture/compare register
- Continuous/One-shot mode
- Selectable software/hardware input trigger
- Programmable digital glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode
- Repetition counter
- Up to 2 independent channels for:
  - Input capture
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on 10 events
- DMA request generation on the following events:
  - Update event
  - Input capture

30.3 LPTIM implementation

Table 281 and *Table 282* describe LPTIM implementation on STM32WBA2 devices.

Table 281. LPTIM device implementation

LPTIM features(1)	LPTIM1	LPTIM2
STM32WBA23	X	X
STM32WBA25	X	X

1. Note ‘X’ = supported, ‘-’ = not supported.

Table 282. LPTIM features

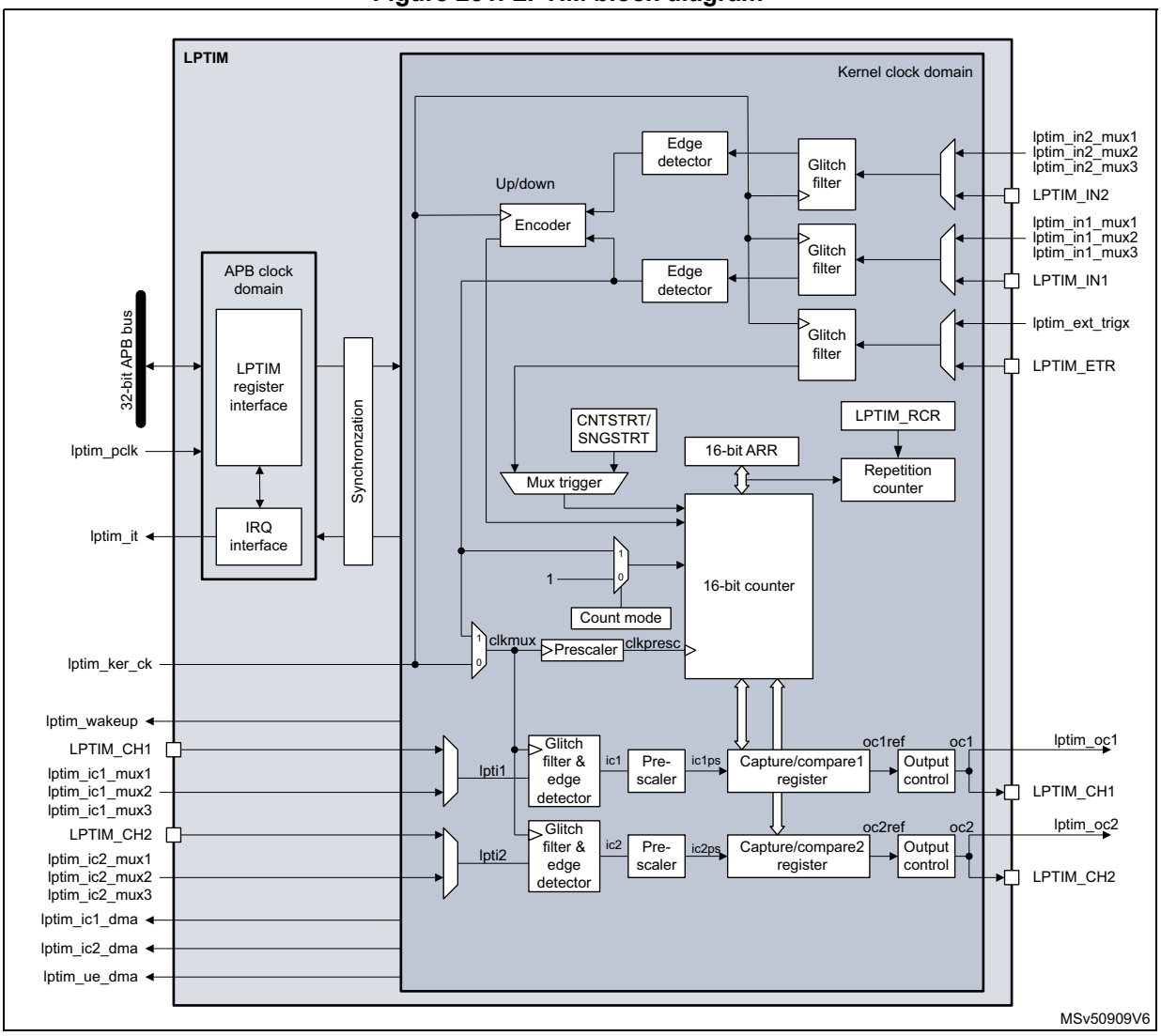
LPTIM modes/features(1)	LPTIM1	LPTIM2
INx	1 to 2	1 to 2
CHx	1 to 2	1 to 2
ETR	X	X
Encoder mode	X	X
PWM mode	X	X
Input Capture	X	X
Number of channels	2	2
Number of DMA requests	3	3
Wake-up from Stop mode	X(2)	X
Autonomous mode	X	X

1. Note ‘X’ = supported, ‘-’ = not supported.
2. Refer to *Section 30.5: LPTIM low-power modes* for more information.

30.4 LPTIM functional description

30.4.1 LPTIM block diagram

Figure 251. LPTIM block diagram(1)



1. Some I/Os may not be available, refer to *Section 30.4.2: LPTIM pins and internal signals*.

30.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

Table 283. LPTIM1/2 input/output pins

Pin name	Pin type	Description
LPTIM_IN1	Digital input	LPTIM Input 1 from GPIO pin on mux input 0
LPTIM_IN2	Digital input	LPTIM Input 2 from GPIO pin on mux input 0

Table 283. LPTIM1/2 input/output pins (continued)

Pin name	Pin type	Description
LPTIM_ETR	Digital input	LPTIM external trigger GPIO pin
LPTIM_CH1	Digital input/output	LPTIM channel 1 input/output GPIO pin
LPTIM_CH2	Digital input/output	LPTIM channel 2 input/output GPIO pin

Table 284. LPTIM1/2 internal signals

Signal name	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1_mux1	Digital input	Internal LPTIM input 1 connected to mux input 1
lptim_in1_mux2	Digital input	Internal LPTIM input 1 connected to mux input 2
lptim_in1_mux3	Digital input	Internal LPTIM input 1 connected to mux input 3
lptim_in2_mux1	Digital input	Internal LPTIM input 2 connected to mux input 1
lptim_in2_mux2	Digital input	Internal LPTIM input 2 connected to mux input 2
lptim_in2_mux3	Digital input	Internal LPTIM input 2 connected to mux input 3
lptim_ic1_mux1	Digital input	Internal LPTIM input capture 1 connected to mux input 1
lptim_ic1_mux2	Digital input	Internal LPTIM input capture 1 connected to mux input 2
lptim_ic1_mux3	Digital input	Internal LPTIM input capture 1 connected to mux input 3
lptim_ic2_mux1	Digital input	Internal LPTIM input capture 2 connected to mux input 1
lptim_ic2_mux2	Digital input	Internal LPTIM input capture 2 connected to mux input 2
lptim_ic2_mux3	Digital input	Internal LPTIM input capture 2 connected to mux input 3
lptim_ext_trigx	Digital input	LPTIM external trigger input x
lptim_it	Digital output	LPTIM global interrupt
lptim_wakeup	Digital output	LPTIM wake-up event
lptim_ic1_dma	Digital output	LPTIM input capture 1 DMA request
lptim_ic2_dma	Digital output	LPTIM input capture 2 DMA request
lptim_ue_dma	Digital output	LPTIM update event DMA request
lptim_ch1	Digital output	LPTIM channel 1 DMA trigger
lptim_ch2	Digital output	LPTIM channel 2 DMA trigger

30.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter.

Table 285. LPTIM1/2 external trigger connections

TRIGSEL	External trigger	
	LPTIM1	LPTIM2
lptim_ext_trig0	LPTIM1_ETR	LPTIM2_ETR
lptim_ext_trig1	rtc_alra_trg	rtc_alra_trg
lptim_ext_trig2	rtc_alrb_trg	rtc_alrb_trg
lptim_ext_trig3	tamp_trg1	tamp_trg1
lptim_ext_trig4	tamp_trg2	gpdma_ch0_tc
lptim_ext_trig5	Reserved	gpdma_ch4_tc
lptim_ext_trig6..7	Reserved	Reserved

Table 286. LPTIM1/2 input 1 connections

lptim_in1_mux	LPTIM1	LPTIM2
lptim_in1_mux0	LPTIM1_IN1	LPTIM2_IN1
lptim_in1_mux1..3	Reserved	Reserved

Table 287. LPTIM1/2 input 2 connections

lptim_in2_mux	LPTIM1	LPTIM2
lptim_in2_mux0	LPTIM1_IN2	LPTIM2_IN2
lptim_in2_mux1..3	Reserved	Reserved

Table 288. LPTIM1/2 input capture 1 connections

lptim_ic1_mux	LPTIM1	LPTIM2
lptim_ic1_mux0	LPTIM1_CH1	LPTIM2_CH1
lptim_ic1_mux1..3	Reserved	Reserved

Table 289. LPTIM1/2 input capture 2 connections

lptim_ic2_mux	LPTIM1	LPTIM2
lptim_ic2_mux0	LPTIM1_CH2	LPTIM2_CH2
lptim_ic2_mux1	LSI	HSI16 / 256
lptim_ic2_mux2	LSE	Reserved
lptim_ic2_mux3	Reserved	Reserved

30.4.4 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be any configurable internal clock source selectable through the RCC (see RCC section for more details). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM can run in one of the following configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM from configurable internal clock source (see RCC section).
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout function or pulse counter function when all the embedded oscillators are turned off after entering a low-power mode.

Programming the CKSEL and COUNTMODE bits allows controlling whether the LPTIM uses an external clock source or an internal one.

When configured to use an external clock source, the CKPOL bits are used to select the external clock signal active edge. If both edges are configured to be active ones, an internal clock signal must also be provided (first configuration). In this case, the internal clock signal frequency must be at least four times higher than the external clock signal frequency.

30.4.5 Glitch filter

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals), are protected with digital filters that prevent any glitches and noise perturbations to propagate inside the LPTIM. This is in order to prevent spurious counts or triggers.

Before activating the digital filters, an internal clock source must first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

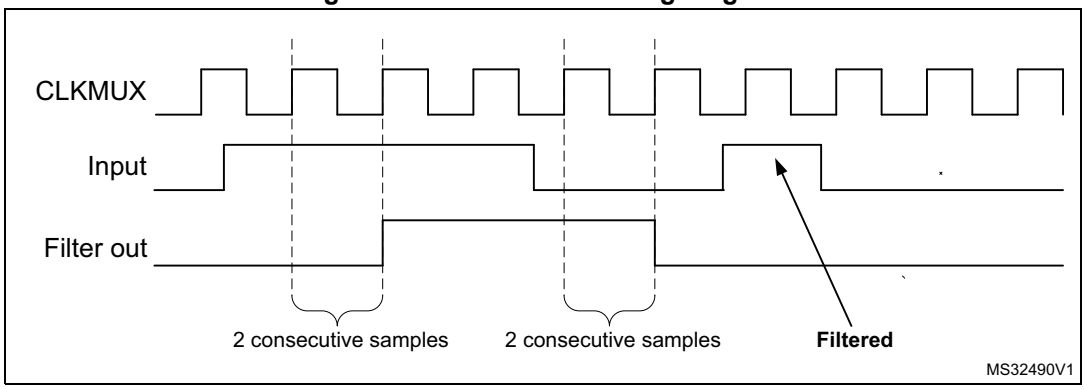
The digital filters are divided into three groups:

- The first group of digital filters protects the LPTIM internal or external inputs. The digital filters sensitivity is controlled by the CKFLT bits
- The second group of digital filters protects the LPTIM internal or external trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.
- The third group of digital filters protects the LPTIM internal or external input captures. The digital filters sensitivity is controlled by the ICxF bits.

Note: The digital filters sensitivity is controlled by groups. It is not possible to configure each digital filter sensitivity separately inside the same group.

The filter sensitivity acts on the number of consecutive equal samples that is detected on one of the LPTIM inputs to consider a signal level change as a valid transition. *Figure 252* shows an example of glitch filter behavior in case of a two consecutive samples programmed.

Figure 252. Glitch filter timing diagram



Note: In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT, ICxF and TRGFLT bits to 0. In this case, an external analog filter can be used to protect the LPTIM external inputs against glitches.

30.4.6 Prescaler

The LPTIM 16-bit counter is preceded by a configurable power-of-2 prescaler. The prescaler division ratio is controlled by the PRESC[2:0] field. The table below lists all the possible division ratios:

Table 290. Prescaler division ratios

Programming	Dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

30.4.7 Trigger multiplexer

The LPTIM counter can be started either by software or after the detection of an active edge on one of the eight trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:

- When TRIGEN[1:0] equals 00, the LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.
- When TRIGEN[1:0] is different than 00, TRIGSEL[2:0] is used to select which of the eight trigger inputs is used to start the counter.

The external triggers are considered asynchronous signals for the LPTIM. After a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it is ignored (unless timeout function is enabled).

Note: The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled is discarded by hardware.

Note: When starting the counter by software (TRIGEN[1:0] = 00), there is a delay of 3 kernel clock cycles between the LPTIM_CR register update (set one of SNGSTRT or CNTSTRT bits) and the effective start of the counter.

30.4.8 Operating mode

The LPTIM features two operating modes:

- Continuous mode: the timer is free running, the timer is started from a trigger event and never stops until the timer is disabled
- One-shot mode: the timer is started from a trigger event and stops when an LPTIM update event is generated.

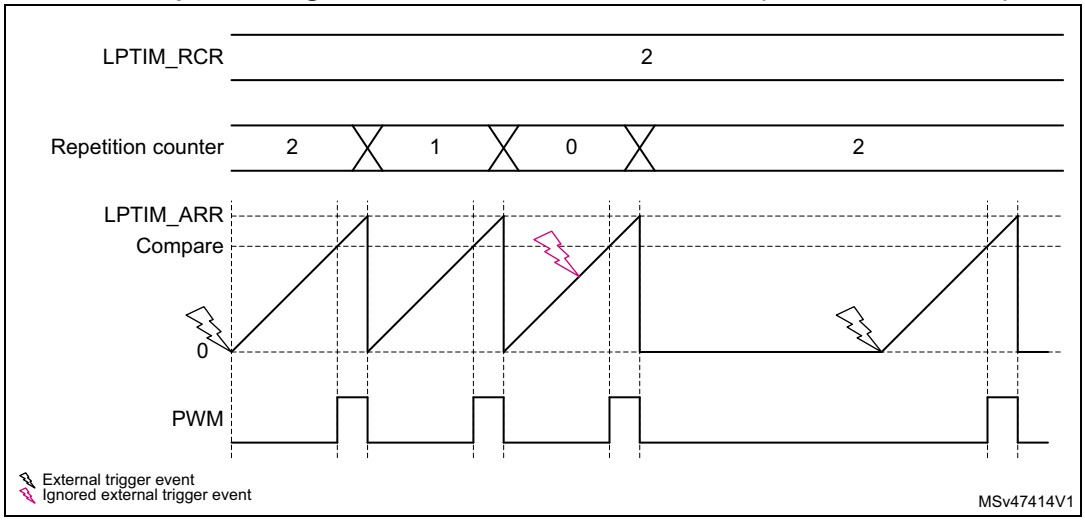
One-shot mode

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event re-starts the timer. Any trigger event occurring after the counter starts and before the next LPTIM update event, is discarded.

In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the repetition counter has stopped (after the update event), and if the repetition register content is different from zero, the repetition counter gets reloaded with the value already contained by the repetition register and a new one-shot counting cycle is started as shown in *Figure 253*.

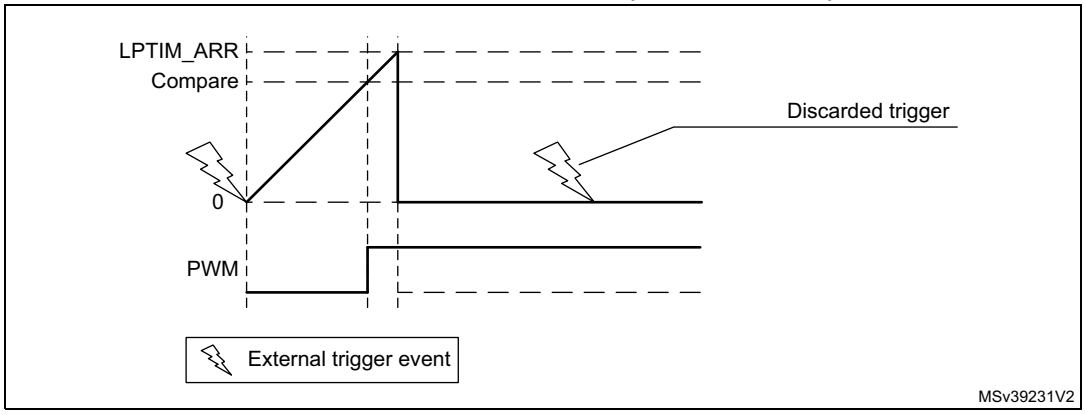
Figure 253. LPTIM output waveform, single-counting mode configuration when repetition register content is different than zero (with PRELOAD = 1)



- Set-once mode activated:

Note that when the WAVE bitfield in the LPTIM_CFGR register is set, the Set-once mode is activated. In this case, the counter is only started once following the first trigger, and any subsequent trigger event is discarded as shown in *Figure 254*.

Figure 254. LPTIM output waveform, single-counting mode configuration and Set-once mode activated (WAVE bit is set)



In case of software start (TRIGEN[1:0] = 00), the SNGSTRT setting starts the counter for one-shot counting.

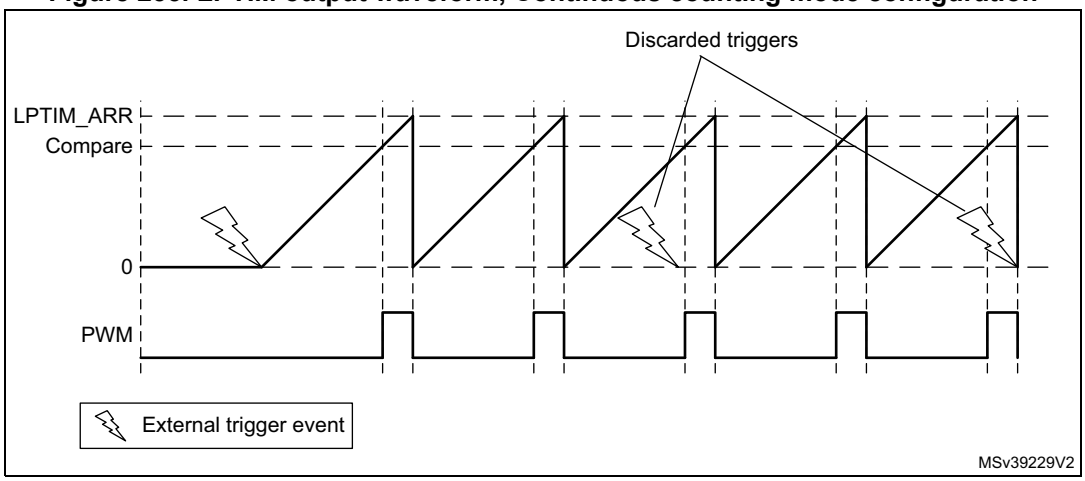
Continuous mode

To enable the continuous counting, the CNTSTRT bit must be set.

In case an external trigger is selected, an external trigger event arriving after CNTSTRT is set, starts the counter for continuous counting. Any subsequent external trigger event is discarded as shown in *Figure 255*.

In case of software start (TRIGEN[1:0] = 00), setting CNTSTRT starts the counter for continuous counting.

Figure 255. LPTIM output waveform, Continuous counting mode configuration



SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (ENABLE bit set to 1). It is possible to change “on the fly” from One-shot mode to Continuous mode.

If the Continuous mode was previously selected, setting SNGSTRT switches the LPTIM to the One-shot mode. The counter (if active) stops as soon as an LPTIM update event is generated.

If the One-shot mode was previously selected, setting CNTSTRT switches the LPTIM to the Continuous mode. The counter (if active) restarts as soon as it reaches ARR.

30.4.9 Timeout function

The detection of an active edge on one selected trigger input can be used to reset the LPTIM counter. This feature is controlled through the TIMOUT bit.

The first trigger event starts the timer, any successive trigger event resets the LPTIM counter and the repetition counter and the timer restarts.

A low-power timeout function can be realized. The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

30.4.10 Waveform generation

Two 16-bit registers, the LPTIM_ARR (autoreload register) and LPTIM_CCRx (capture/compare register), are used to generate several different waveforms on LPTIM output

The timer can generate the following waveforms:

- The PWM mode: the LPTIM output is set as soon as the counter value in LPTIM_CNT exceeds the compare value in LPTIM_CCRx. The LPTIM output is reset as soon as a match occurs between the LPTIM_ARR and the LPTIM_CNT register. For more details see *Section 30.4.19: PWM mode*.
- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require the LPTIM_ARR register value to be strictly greater than the LPTIM_CCRx register value.

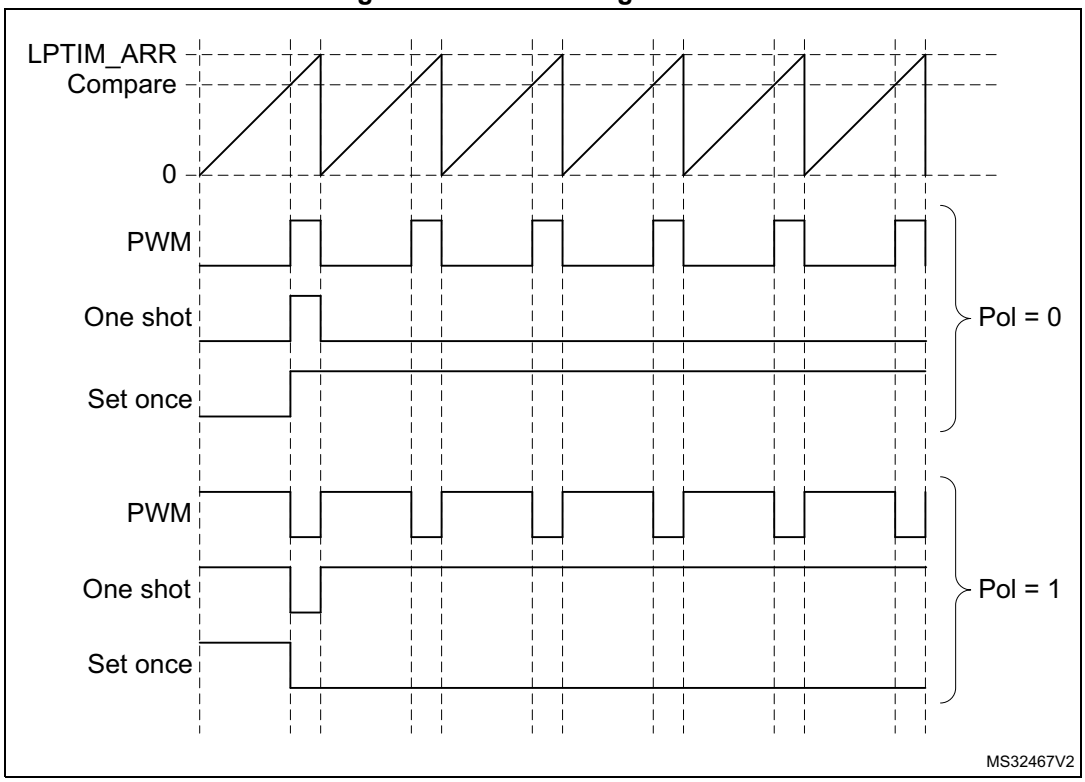
The LPTIM output waveform can be configured through the WAVE bit as follow:

- Resetting the WAVE bit to 0 forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTRT or SNGSTRT.
- Setting the WAVE bit to 1 forces the LPTIM to generate a Set-once mode waveform.

The CCxP bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value changes immediately after the polarity is re-configured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by two can be generated. *Figure 256* below shows the three possible waveforms that can be generated on the LPTIM output. Also, it shows the effect of the polarity change using the CCxP bit.

Figure 256. Waveform generation



30.4.11 Register update

The LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register are updated immediately after the APB bus write operation or in synchronization with the next LPTIM update event if the timer is already started.

The PRELOAD bit controls how the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated:

- When the PRELOAD bit is reset to 0, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are immediately updated after any write access.
- When the PRELOAD bit is set to 1, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated at next LPTIM update event, if the timer has been already started.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the APB write and the moment when these values are available to the counter comparator. Within this latency period, any additional write into these registers must be avoided.

The ARROK flag, the REPOK flag and the CMPxOK flag in the LPTIM_ISR register indicate when the write operation is completed to respectively the LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register.

After a write to the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before respectively the ARROK flag, the REPOK flag or the CMPxOK flag be set, leads to unpredictable results.

30.4.12 Counter mode

The LPTIM counter can be used to count external events on the LPTIM input1 or it can be used to count internal clock cycles. The CKSEL and COUNTMODE bits control which source is used for updating the counter.

In case the LPTIM is configured to count external events on Input1, the counter can be updated following a rising edge, falling edge or both edges depending on the value written to the CKPOL[1:0] bits.

The count modes below can be selected, depending on CKSEL and COUNTMODE values:

- CKSEL = 0: the LPTIM is clocked by an internal clock source
  - COUNTMODE = 0

    The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - COUNTMODE = 1

    The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM.

    Consequently, in order not to miss any event, the frequency of the changes on the external Input1 signal must never exceed the frequency of the internal clock provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRESC[2:0] = 000).
- CKSEL = 1: the LPTIM is clocked by an external clock source

  COUNTMODE value is don't care.

  In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external input1 is used as system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.

  For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.

  Since the signal injected on the LPTIM external input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

30.4.13 Timer enable

The ENABLE bit located in the LPTIM_CR register is used to enable/disable the LPTIM kernel logic. After setting the ENABLE bit, a delay of two counter clock cycles is needed before the LPTIM is actually enabled.

The LPTIM_CFGR register must be modified only when the LPTIM is disabled.

30.4.14 Timer counter reset

In order to reset the content of LPTIM_CNT register, two reset mechanisms are implemented:

- The synchronous reset mechanism: the synchronous reset is controlled by the COUNTRST bit in the LPTIM_CR register. After setting the COUNTRST bitfield to '1', the reset signal is propagated in the LPTIM kernel clock domain. So it is important to note that a few clock pulses of the LPTIM kernel logic elapse before the reset is taken into account. This makes the LPTIM counter count few extra pluses between the time when the reset is triggered and it become effective. Since the COUNTRST bit is located in the APB clock domain and the LPTIM counter is located in the LPTIM kernel clock domain, a delay of 3 clock cycles of the kernel clock is needed to synchronize the reset signal issued by the APB clock domain when writing '1' to the COUNTRST bit.

Note: The software ensures that COUNRST bit is '0' before generating every synchronous reset.

- The asynchronous reset mechanism: the asynchronous reset is controlled by the RSTARE bit located in the LPTIM_CR register. When this bit is set to '1', any read access to the LPTIM_CNT register resets its content to zero. Asynchronous reset must be triggered within a timeframe in which no LPTIM core clock is provided. For example when LPTIM Input1 is used as external clock source, the asynchronous reset must be applied only when there is enough insurance that no toggle occurs on the LPTIM Input1.
  To read reliably the content of the LPTIM_CNT register two successive read accesses must be performed and compared. A read access can be considered reliable when the value of the two read accesses is equal. Unfortunately when asynchronous reset is enabled there is no possibility to read twice the LPTIM_CNT register.

Warning: There is no mechanism inside the LPTIM that prevents the two reset mechanisms from being used simultaneously. The developer must make sure that these two mechanisms are used exclusively.

30.4.15 Encoder mode

This mode allows handling signals from quadrature encoders used to detect angular position of rotary elements. Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value programmed into the LPTIM_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore LPTIM_ARR must be configured before starting the counter. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signalized by the two down and up flags in the LPTIM_ISR register. An interrupt can be generated for both direction change events if enabled through the DOWNIE bit.

To activate the Encoder mode the ENC bit has to be set to 1. The LPTIM must first be configured in Continuous mode.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder's position. The count direction, signaled by the Up and Down flags, correspond to the rotation direction of the encoder rotor.

According to the edge sensitivity configured using the CKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

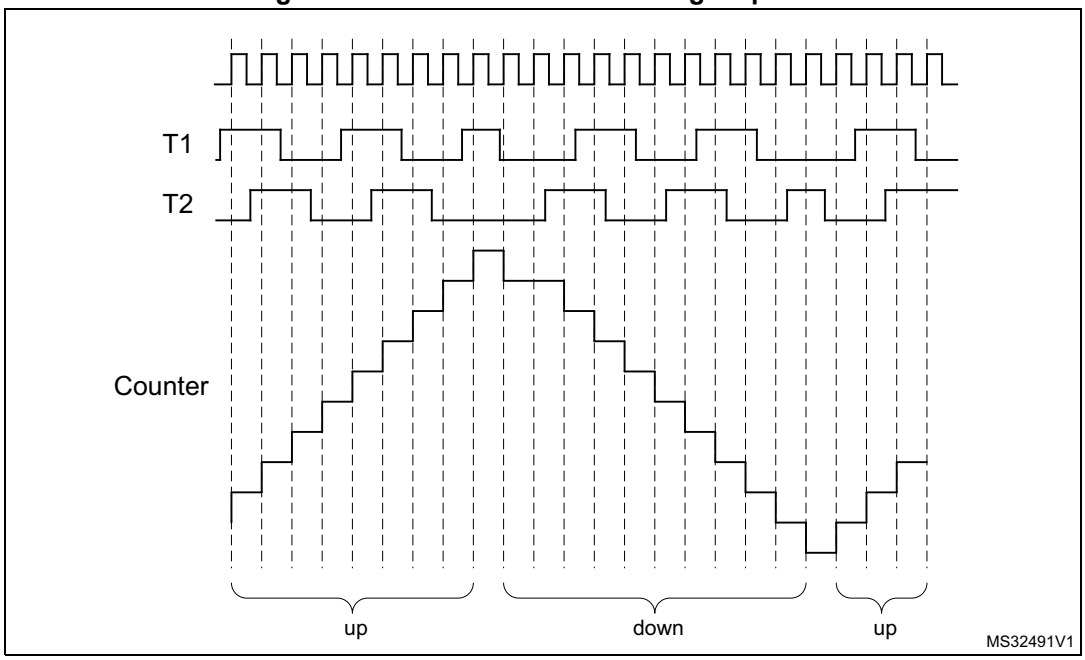
Table 291. Encoder counting scenarios

Active edge	Level on opposite signal (Input1 for Input2, Input2 for Input1)	Input1 signal		Input2 signal	
		Rising	Falling	Rising	Falling
Rising Edge (Encoder sub-mode 1)	High	Down	No count	Up	No count
	Low	Up	No count	Down	No count
Falling Edge (Encoder sub-mode 2)	High	No count	Up	No count	Down
	Low	No count	Down	No count	Up
Both Edges (Encoder sub-mode 3)	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

The following figure shows a counting sequence for Encoder mode where both-edge sensitivity is configured.

Caution: In this mode the LPTIM must be clocked by an internal clock source, so the CKSEL bit must be maintained to its reset value which is equal to 0. Also, the prescaler division ratio must be equal to its reset value which is 1 (PRESC[2:0] bits must be 000).

Figure 257. Encoder mode counting sequence



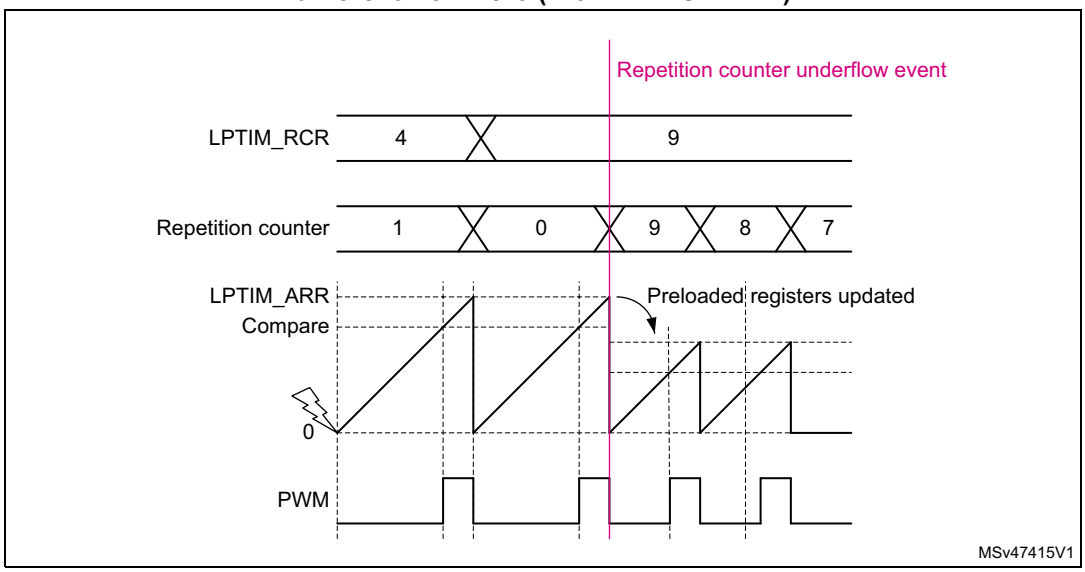
30.4.16 Repetition counter

The LPTIM features a repetition counter that decrements by 1 each time an LPTIM counter overflow event occurs. A repetition counter underflow event is generated when the repetition counter contains zero and the LPTIM counter overflows. Next to each repetition counter underflow event, the repetition counter gets loaded with the content of the REP[7:0] bitfield which belongs to the repetition register LPTIM_RCR.

A repetition underflow event is generated on each and every LPTIM counter overflow when the REP[7:0] register is set to 0.

When PRELOAD = 1, writing to the REP[7:0] bitfield has no effect on the content of the repetition counter until the next repetition underflow event occurs. The repetition counter continues to decrement each LPTIM counter overflow event and only when a repetition underflow event is generated, the new value written into REP[7:0] is loaded into the repetition counter. This behavior is depicted in *Figure 258*.

Figure 258. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1)



A repetition counter underflow event is systematically associated with LPTIM preloaded registers update (refer to *Section 30.4.11: Register update* for more information).

Repetition counter underflow event is signaled to the software through the update event (UE) flag mapped into the LPTIM_ISR register. When set, the UE flag can trigger an LPTIM interrupt if its respective update event interrupt enable (UEIE) control bit, mapped to the LPTIM_DIER register, is set.

The repetition register LPTIM_RCR is located in the APB bus interface clock domain where the repetition counter itself is located in the LPTIM kernel clock domain. Each time a new value is written to the LPTIM_RCR register, this new content is propagated from the APB bus interface clock domain to the LPTIM kernel clock domain. The new written value is then loaded to the repetition counter immediately after a repetition counter underflow event. The synchronization delay for the new written content is four APB clock cycles plus three LPTIM kernel clock cycles and it is signaled by the REPOK flag located in the LPTIM_ISR register when it is elapsed. When the LPTIM kernel clock cycle is relatively slow, for instance when the LPTIM kernel is being clocked by the LSI clock source, it can be lengthy to keep polling on the REPOK flag by software to detect that the synchronization of the LPTIM_RCR register content is finished. For that reason, the REPOK flag, when set, can generate an interrupt if its associated REPOKIE control bit in the LPTIM_DIER register is set.

Note: After a write to the LPTIM_RCR register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive writes before the REPOK flag is set, lead to unpredictable results.

Caution: When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the autoreload match event, otherwise an unpredictable behavior may occur.

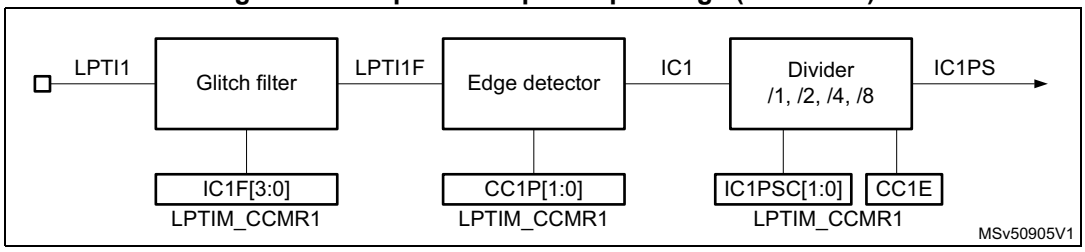
30.4.17 Capture/compare channels

Each capture/compare channel is built around a capture/compare register, an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control) for PWM.

Input stage

The input stage samples the corresponding LPTIx input to generate a filtered signal LPTIxF. Then, an edge detector with polarity selection generates ICx signal used as the capture command. It is prescaled to generate the capture command signal (ICxPS).

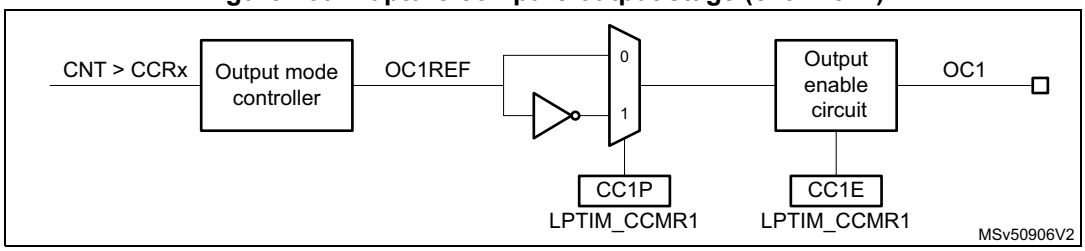
Figure 259. Capture/compare input stage (channel 1)



Output stage

The output stage generates an intermediate waveform which is then used for reference: OCxREF (active high). The polarity acts at the end of the chain.

Figure 260. Capture/compare output stage (channel 1)



30.4.18 Input capture mode

In Input capture mode, the capture/compare registers (LPTIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. Assuming input capture is enabled on a channel x (CCxE set) and when a capture occurs, the corresponding CCxIF flag (LPTIM_ISR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (LPTIM_ISR register) is set. CCxIF can be cleared by software by writing the CCxICF to 1 or by reading the captured data stored in the LPTIM_CCRx register. CCxOF is cleared by writing CCxOCF to 1.

Note: In DMA mode, the input capture channel have to be enabled (set CCxE bit) the last, after enabling the DMA request and after starting the counter. This is in order to prevent generating an input capture DMA request when the counter is not started yet.

Input capture Glitch filter latency

When a trigger event arrives on channel x input (LPTIx) and depending on the configured glitch filter (ICxF[1:0] field in CCMRx register) and on the kernel clock prescaler value (PRESC[2:0] field in CFGR register), there is a variable latency that leads to a systematic offset (see *Table 292*) between the captured value stored in the CCRx register and the real value corresponding to the capture trigger.

This offset has no impact on pulse width measurement as it is systematic and compensated between two captures.

The real capture value corresponding to the input capture trigger can be calculated using the below formula:

Real capture value = captured(LPTIM_CCRx) - offset

The relevant offset must be used depending on the glitch filter and on the kernel clock prescaler value (PRESC field in CFGR register)

Example: determining the real capture value when PRESC[2:0] = 0x2 and ICxF = 0x3.

For this configuration (PRESC[2:0] = 0x2 and ICxF = 0x3) and according to the *Table 292*, the offset is 5.

Assuming that the captured value in CCRx is 9 (LPTIM_CNT = 9), this means that the capture trigger occurred when the LPTIM_CNT was equal to 9 - 5 = 4.

Table 292. Input capture Glitch filter latency (in counter step unit)

Prescaler PRESC[2:0]	ICxF[1:0]	Offset
0	0	2
	1	7
	2	9
	3	13
1	0	3
	1	5
	2	6
	3	8
2	0	2
	1	3
	2	4
	3	5
3	0	2
	1	2
	2	3
	3	3
4	0	2
	1	2
	2	2
	3	2
5	0	2
	1	2
	2	2
	3	2

Table 292. Input capture Glitch filter latency (in counter step unit) (continued)

Prescaler PRESC[2:0]	ICxF[1:0]	Offset
6	0	2
	1	2
	2	2
	3	2
7	0	2
	1	2
	2	2
	3	2

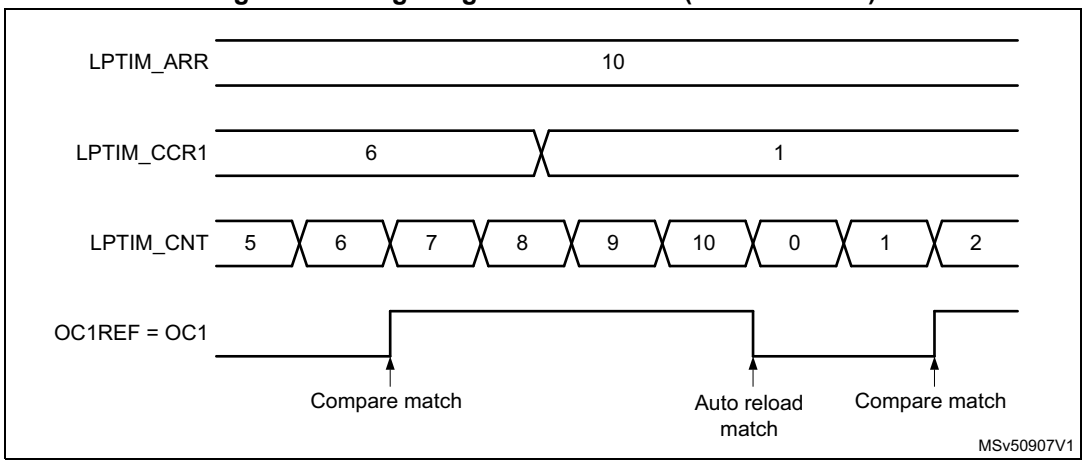
30.4.19 PWM mode

The PWM mode enables to generate a signal with a frequency determined by the value of the LPTIM_ARR register and a duty cycle determined by the value of the LPTIM_CCRx register. The LPTIM is able to generate PWM in edge-aligned mode.

OCx polarity is software programmable using the CCxP bit in the LPTIM_CCMRx register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the LPTIM_CCMRx register. Refer to the LPTIM_CCMRx register description for more details.

Figure 261 gives an example where the LPTIM channel 1 is configured in PWM mode with LPTIM_CCR1 = 6 then 1 and LPTIM_ARR=10.

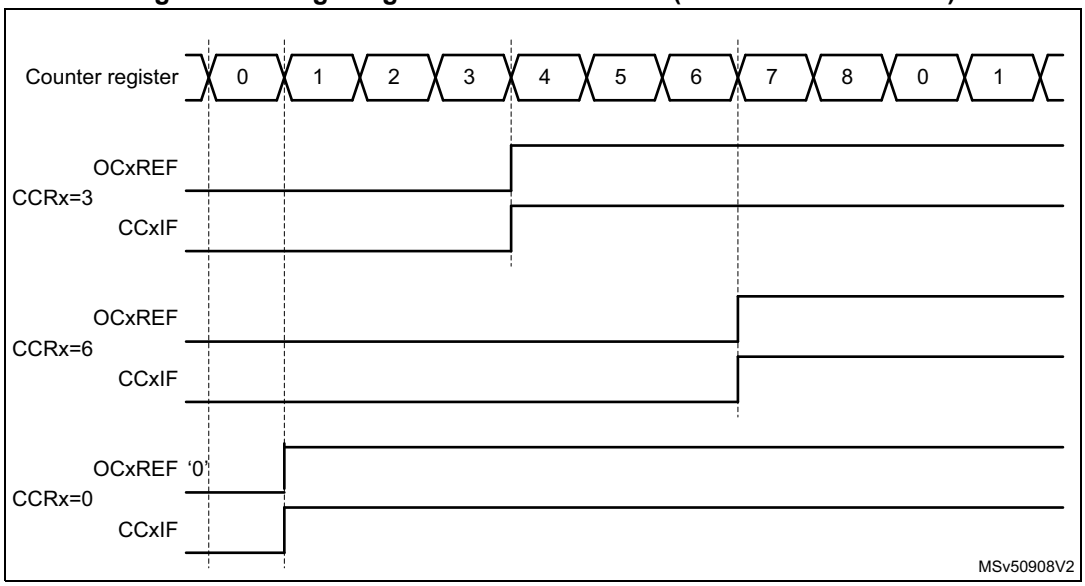
Figure 261. Edge-aligned PWM mode (PRELOAD = 1)



In the following example the reference PWM signal OCxREF is low as long as LPTIM_CNT ≤ LPTIM_CCRx else it becomes high.

Figure 262 shows some edge-aligned PWM waveforms in an example where LPTIM_ARR = 8.

Figure 262. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0)



PWM mode with immediate update PRELOAD = 0

The PWM mode with PRELOAD = 0 enables the early change of the output level within the current PWM cycle. Based on the immediate update (PRELOAD = 0) of the LPTIM_CCRx register and on the continuous comparison of LPTIM_CNT and LPTIM_CCRx registers, it permits to have a new duty cycle value applied as soon as possible within the current PWM cycle, without having to wait for the completion of the current PWM period.

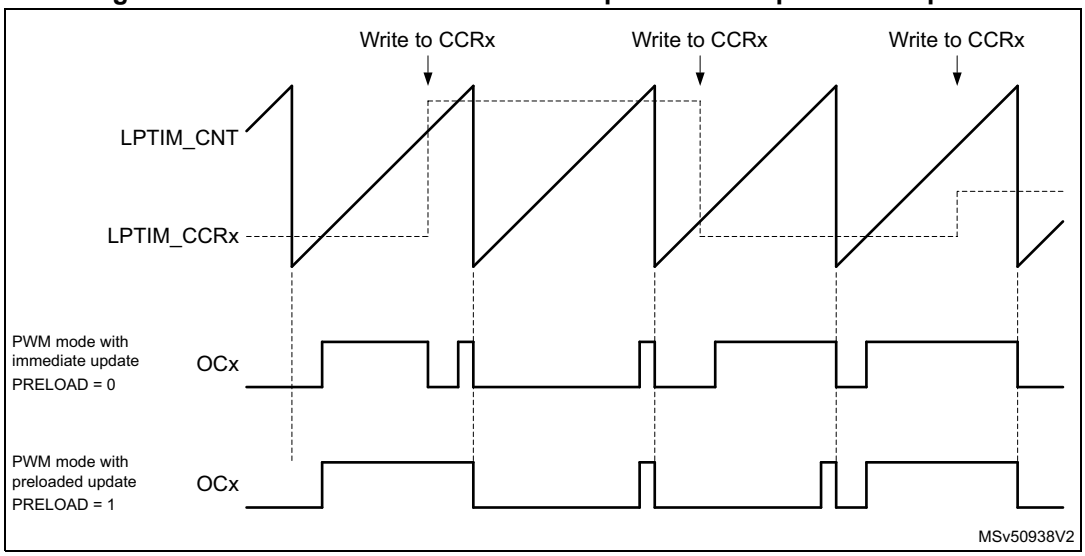
When the (PRELOAD = 0), the OCxREF signal level can be changed on-the-fly by software (or DMA) by updating the compare value in the LPTIM_CCRx register.

Depending on the written compare value and on the current counter and compare values, the OCxREF level is re-assigned as illustrated below:

- If the new compare value does not exceed the current counter value and the current compare value exceeds the counter, OCxREF level is re-assigned high as soon as the new compare value is written.
- If the new compare value exceeds the counter value and the current compare value does not exceed the counter, OCxREF level is re-assigned low as soon as the new compare value is written.

The output reference signal OCxREF level is left unchanged when none of the new compare value and the current compare value exceed the counter. *Figure 263* illustrates the behavior of the OCxREF signal level when PRELOAD = 0 and PRELOAD = 1.

Figure 263. PWM mode with immediate update versus preloaded update



Note: For both PWM modes, the compare match, auto-reload match, and the update event flags are set one LPTIM counter cycle later after the corresponding event, the OCxREF level is also changed one LPTIM counter cycle later after the corresponding event. For instance when the LPTIM_CCRx is set to 3 the CCxIF is set when the LPTIM_CNT = 4. Figure 261 illustrates this behavior.

30.4.20 Autonomous mode

When clocked by oscillators available in this mode (refer to RCC), the LPTIM can operate in autonomous mode, remaining then fully functional in Stop mode where the APB clock is stopped. The APB clock is requested by the peripheral each time a data must be transferred from or to the SRAM. Once the APB clock is received by the peripheral, either an interrupt or a DMA request is generated, depending on the LPTIM configuration.

In order to offload the CPU (in Run mode) or to avoid to wake it up when in Stop mode, it is possible to use LPTIM DMA requests to transfer the captured values when in input capture mode or to update LPTIM registers when in PWM mode.

When in Stop mode, the LPTIM counter can be automatically started after the detection of an active edge on one of its external input triggers.

Input capture mode

To operate autonomously in Stop mode, the input capture DMA request must be enabled by setting the CCxDE bit in the LPTIM_DIER register.

Each time a counter value is captured and available in the LPTIM_CCRx register, the APB clock is requested by the peripheral and a DMA request is generated. The captured value is then transferred to the SRAM. The CCxIF flag is automatically cleared by hardware once the captured value is read by APB (can be any bus master like CPU or DMA).

PWM mode

The LPTIM can be configured to autonomously change, at each update event, the output waveform pulse width and/or the duty cycle without any CPU intervention. To enable this autonomous mode, the corresponding UEDE bit must be set in the LPTIM_DIER register.

At each update event, the APB clock is requested by the peripheral and a DMA request is generated. DMA direction must be configured as memory-to-peripheral which enables updating LPTIM registers, at each DMA request, with values stored in SRAM.

The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA. Thus, to enable automatic hardware clearing of UE flag, the application must configure the LPTIM_ARR register to be the last one to be written (at the end of list). For instance, if LPTIM_CCR1 and LPTIM_RCR registers need to be updated in Stop mode by DMA, the update sequence must be: LPTIM_CCR1, LPTIM_RCR then LPTIM_ARR.

The UE flag can also be cleared over its corresponding clear bit UECF in the LPTIM_ICR register, this can be done by configuring the DMA to write the LPTIM_ICR register at the end of register update.

30.4.21 DMA requests

The LPTIM can generate two categories of DMA requests:

- DMA requests used to retrieve the input-capture counter values
- DMA update requests used to re-program part of the LPTIM, multiple times, at regular intervals, without software overhead

Input capture DMA request

Each LPTIM channel has its dedicated input capture DMA request. A DMA request is generated (if CCxDE bit is set in LPTIM_DIER) and CCxIF is set each time a capture is ready in the LPTIM_CCRx register. The captured values in LPTIM_CCRx can then be transferred regularly by DMA to the desired memory destination. The CCxIF is automatically cleared by hardware when the captured value in LPTIM_CCRx register is read.

Note: The ICx DMA request signal lptim_icx_dma is reset in the following conditions:

- *if the corresponding DMA request is disabled (clear CCxDE bit in the LPTIM_DIER register)*
- *or if the channel x is disabled (clear CCxE bit)*
- *or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)*

Update event DMA request

A DMA request is generated (if UEDE is set in LPTIM_DIER) and the UE flag is set at each update event. DMA request can be used to regularly update the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx registers permitting to generate custom PWM waveforms.

The UE is automatically cleared by hardware upon any bus master (like CPU or DMA) write access to the LPTIM_ARR register.

Note: The UE DMA request signal lptim_ue_dma is reset in the following conditions:

- *if the corresponding DMA request is disabled (clear UEDE bit in the LPTIM_DIER register)*
- *or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)*
- *or if the channel x is disabled (clear CCxE bit) and all the other channels are already disabled*

30.4.22 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the timer dedicated bit configuration in the debug support (DBG) peripheral.

For further details, refer to section debug support (DBG).

30.5 LPTIM low-power modes

Table 293. Effect of low-power modes on the LPTIM

Mode	LPTIMx	Description
Sleep	1, 2	No effect. LPTIM interrupts cause the device to exit Sleep mode.
Stop 0 and Stop 1	1, 2	If the LPTIM is clocked by an oscillator available in Stop mode, LPTIM is functional and the interrupts cause the device to exit the Stop mode. The DMA requests are functional if the instance supports the autonomous mode (refer to *Section 30.3: LPTIM implementation*).
Stop 2	1	If the LPTIM is clocked by an oscillator available in Stop mode, LPTIM is functional and the interrupts cause the device to exit the Stop mode. The DMA requests are not functional.
	2	The LPTIM peripheral is powered down and must be reinitialized after exiting Stop 2 mode.
Stop 3	1	The content of the LPTIM registers is kept. No autonomous mode.
	2	The LPTIM peripheral is powered down and must be reinitialized after exiting Stop 3 mode.
Standby	1, 2	The LPTIM peripheral is powered down and must be reinitialized after exiting Standby mode.

30.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM_DIER register:

- Compare match
- Auto-reload match (whatever the direction if Encoder mode)
- External trigger event
- Autoreload register write completed
- Compare register write completed
- Direction change (Encoder mode), programmable (up / down / both).
- Update Event
- Repetition register update OK
- Input capture occurred
- Over-capture occurred
- Interrupt enable register update OK

Note: If any bit in the LPTIM_DIER register is set after that its corresponding flag in the LPTIM_ISR register (status register) is set, the interrupt is not asserted.

Table 294. Interrupt events

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop mode(1)
LPTIMx	Compare match	CCxIF	CCxIE	Write 1 to CCxCF	Yes	Yes
	Input capture	CCxIF	CCxIE	Write 1 to CCxCF	Yes	Yes
	Over-capture	CCxOF	CCxOIE	Write 1 to CCxOCF	Yes	Yes
	Auto-reload match	ARRM	ARRMIE	Write 1 to ARRMCF	Yes	Yes
	External trigger event	EXTTRIG	EXTTRIGIE	Write 1 to EXTTRIGCF	Yes	Yes
	Auto-reload register update OK	ARROK	ARROKIE	Write 1 to ARROKCF	Yes	Yes
	Capture/compare register update OK	CMPxOK	CMPxOKIE	Write 1 to CMPxOKCF	Yes	Yes
	Direction change to up(2)	UP	UPIE	Write 1 to UPCF	Yes	Yes
	Direction change to down(2)	DOWN	DOWNIE	Write 1 to DOWNCF	Yes	Yes
	Update event	UE	UEIE	Write 1 to UECF	Yes	Yes
	Repetition register update OK	REPOK	REPOKIE	Write 1 to REPOKCF	Yes	Yes

1. Each LPTIM event can wake up the device from Stop mode only if the LPTIM instance supports the wake-up from Stop mode feature. Refer to *Section 30.3: LPTIM implementation*.
2. If LPTIM does not support Encoder mode feature, this event does not exist. Refer to *Section 30.3: LPTIM implementation*.

30.7 LPTIM registers

Refer to *Section 1.2: List of abbreviations for registers on page 66* for a list of abbreviations used in register descriptions.

The peripheral registers can only be accessed by words (32-bit).

30.7.1 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1, 2)

This description of the register can only be used for output compare mode. See next section for input capture mode.

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	CMP2OK	Res.	Res.	Res.
							r					r			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2IF	REPOK	UE	DOWN	UP	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
						r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OK**: Compare register 2 update OK

CMP2OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR2 register has been successfully completed. CMP2OK flag can be cleared by writing 1 to the CMP2OKCF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 30.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Compare 2 interrupt flag

If channel CC2 is configured as output:

The CC2IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR2 register's value

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 30.3.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 30.3.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 30.3.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 **CMP1OK**: Compare register 1 update OK

CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF bit in the LPTIM_ICR register.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: Compare 1 interrupt flag

If channel CC1 is configured as output:

The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register's value

30.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1, 2)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2OF	CC1OF	Res.	Res.	CC2IF	REPOK	UE	DOWN	UP	ARROK	Res.	EXTTRIG	ARRM	CC1IF
		r	r			r	r	r	r	r	r		r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OF**: Capture 2 over-capture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC2OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR2 register while CC2IF flag was already set.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 30.3.

Bit 12 **CC1OF**: Capture 1 over-capture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC1OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR1 register while CC1IF flag was already set.

Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 30.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Capture 2 interrupt flag

If channel CC2 is configured as input:

CC2IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR2 register. The corresponding interrupt or DMA request is generated if enabled. The CC2OF flag is set if the CC2IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR2 register. (An edge has been detected on IC2 which matches the selected polarity). The CC2IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA). The CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 30.3.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. The UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 30.3.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 30.3.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: capture 1 interrupt flag

If channel CC1 is configured as input:

CC1IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR1 register. The corresponding interrupt or DMA request is generated if enabled. The CC1OF flag is set if the CC1IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR1 register. (An edge has been detected on IC1 which matches the selected polarity). The CC1IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA).CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

30.7.3 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1, 2)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OKCF	Res.	Res.	Res.	Res.	CMP2 OKCF	Res.	Res.	Res.
							w					w			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2CF	REPOK CF	UECF	DOWN CF	UPCF	ARR OKCF	CMP1 OKCF	EXT TRIG CF	ARRM CF	CC1CF
						w	w	w	w	w	w	w	w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag

Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OKCF**: Compare register 2 update OK clear flag

Writing 1 to this bit clears the CMP2OK flag in the LPTIM_ISR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 30.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2CF**: Capture/compare 2 clear flag
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 30.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 30.3.

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 30.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMP1OKCF**: Compare register 1 update OK clear flag
Writing 1 to this bit clears the CMP1OK flag in the LPTIM_ISR register.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

30.7.4 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1, 2)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2 OCF	CC1 OCF	Res.	Res.	CC2CF	REPOK CF	UECF	DOWN CF	UPCF	ARRO KCF	Res.	EXTTR IGCF	ARRM CF	CC1CF
		w	w			w	w	w	w	w	w		w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OCF**: Capture/compare 2 over-capture clear flag
Writing 1 to this bit clears the CC2OF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 30.3.

Bit 12 **CC1OCF**: Capture/compare 1 over-capture clear flag
Writing 1 to this bit clears the CC1OF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 30.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2CF**: Capture/compare 2 clear flag
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 30.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 30.3.

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 30.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

30.7.5 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1, 2)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UEDE	Res.	Res.	Res.	CMP2OKIE	Res.	Res.	Res.
								rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2IE	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **UEDE**: Update event DMA request enable
0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
1: UE DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 30.3.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OKIE**: Compare register 2 update OK interrupt enable
0: CMPOK register 2 interrupt disabled
1: CMPOK register 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 30.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable
0: Capture/compare 2 interrupt disabled
1: Capture/compare 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 30.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 30.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 30.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 **CMP1OKIE**: Compare register 1 update OK interrupt enable
0: CMPOK register 1 interrupt disabled
1: CMPOK register 1 interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

30.7.6 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1, 2)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CC2DE	Res.	UEDE	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE
						rw		rw							rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2OIE	CC1OIE	Res.	Res.	CC2IE	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	Res.	EXTTRIGIE	ARRMIE	CC1IE
		rw	rw			rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 Reserved, must be kept at reset value.

Bit 25 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled. Writing '0' to the CC2DE bit resets the associated ic2_dma_req signal.
1: CC2 DMA request enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 30.3.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **UEDE**: Update event DMA request enable
0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
1: UE DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 30.3.

Bits 22:17 Reserved, must be kept at reset value.

Bit 16 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled. Writing '0' to the CC1DE bit resets the associated ic1_dma_req signal.
1: CC1 DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 30.3.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OIE**: Capture/compare 2 over-capture interrupt enable
0: CC2 over-capture interrupt disabled
1: CC2 over-capture interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 30.3.

Bit 12 **CC1OIE**: Capture/compare 1 over-capture interrupt enable
0: CC1 over-capture interrupt disabled
1: CC1 over-capture interrupt enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 30.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable
0: Capture/compare 2 interrupt disabled
1: Capture/compare 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 30.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 30.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 30.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

30.7.7 LPTIM configuration register (LPTIM_CFGR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC	COUNT MODE	PRE LOAD	Res.	WAVE	TIMOUT	TRIGEN[1:0]		Res.
							rw	rw	rw		rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGSEL[2:0]			Res.	PRESC[2:0]			Res.	TRGFLT[1:0]		Res.	CKFLT[1:0]		CKPOL[1:0]		CKSEL
rw	rw	rw		rw	rw	rw		rw	rw		rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **ENC**: Encoder mode enable
The ENC bit controls the Encoder mode
0: Encoder mode disabled
1: Encoder mode enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 30.3.

Bit 23 **COUNTMODE**: counter mode enabled
The COUNTMODE bit selects which clock source is used by the LPTIM to clock the counter:
0: The counter is incremented following each internal clock pulse
1: The counter is incremented following each valid clock pulse on the LPTIM external Input1

Bit 22 **PRELOAD**: Registers update mode
The PRELOAD bit controls the LPTIM_ARR, LPTIM_RCR and the LPTIM_CCRx registers update modality
0: Registers are updated after each APB bus write access
1: Registers are updated at the end of the current LPTIM period

Bit 21 Reserved, must be kept at reset value.

Bit 20 **WAVE**: Waveform shape
The WAVE bit controls the output shape
0: Deactivate Set-once mode
1: Activate the Set-once mode

Bit 19 **TIMOUT**: Timeout enable
The TIMOUT bit controls the Timeout feature
0: A trigger event arriving when the timer is already started is ignored
1: A trigger event arriving when the timer is already started resets and restarts the LPTIM counter and the repetition counter

Bits 18:17 **TRIGEN[1:0]**: Trigger enable and polarity
The TRIGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:
00: Software trigger (counting start is initiated by software)
01: Rising edge is the active edge
10: Falling edge is the active edge
11: Both edges are active edges

Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **TRIGSEL[2:0]**: Trigger selector
The TRIGSEL bits select the trigger source that serves as a trigger event for the LPTIM among the below 8 available sources:
000: lptim_ext_trig0
001: lptim_ext_trig1
010: lptim_ext_trig2
011: lptim_ext_trig3
100: lptim_ext_trig4
101: lptim_ext_trig5
110: lptim_ext_trig6
111: lptim_ext_trig7
See *Section 30.4.3: LPTIM input and trigger mapping* for details.

Bit 12 Reserved, must be kept at reset value.

Bits 11:9 **PRESC[2:0]:** Clock prescaler
The PRESC bits configure the prescaler division factor. It can be one among the following division factors:
000: /1
001: /2
010: /4
011: /8
100: /16
101: /32
110: /64
111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **TRGFLT[1:0]:** Configurable digital filter for trigger

The TRGFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: Any trigger active level change is considered as a valid trigger

01: Trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.

10: Trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.

11: Trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **CKFLT[1:0]:** Configurable digital filter for external clock

The CKFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: Any external clock signal level change is considered as a valid transition

01: External clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: External clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: External clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 2:1 **CKPOL[1:0]:** Clock polarity

When the LPTIM is clocked by an external clock source, CKPOL bits is used to configure the active edge or edges used by the counter:

00: The rising edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 1 is active.

01: The falling edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 2 is active.

10: Both edges are active edges. When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four times the external clock frequency.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 3 is active.

11: Not allowed

Refer to *Section 30.4.15: Encoder mode* for more details about Encoder sub-modes.

Bit 0 **CKSEL:** Clock selector

The CKSEL bit selects which clock source the LPTIM uses:

0: LPTIM is clocked by internal clock source (APB clock or any of the embedded oscillators)

1: LPTIM is clocked by an external clock source through the LPTIM external Input1

Caution: The LPTIM_CFGR register must only be modified when the LPTIM is disabled (ENABLE bit reset to 0).

30.7.8 LPTIM control register (LPTIM_CR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
											rw	rs	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RSTARE:** Reset after read enable

This bit is set and cleared by software. When RSTARE is set to '1', any read access to LPTIM_CNT register asynchronously resets LPTIM_CNT register content.

This bit can be set only when the LPTIM is enabled.

Bit 3 **COUNTRST:** Counter reset

This bit is set by software and cleared by hardware. When set to '1' this bit triggers a synchronous reset of the LPTIM_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock can be different from APB clock).

This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Caution: COUNTRST must never be set to '1' by software before it is already cleared to '0' by hardware. Software must consequently check that COUNTRST bit is already cleared to '0' before attempting to set it to '1'.

Bit 2 **CNTSTRT:** Timer start in Continuous mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = 00), setting this bit starts the LPTIM in Continuous mode.

If the software start is disabled (TRIGEN[1:0] different than 00), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.

If this bit is set when a single pulse mode counting is ongoing, then the timer does not stop at the next match between the LPTIM_ARR and LPTIM_CNT registers and the LPTIM counter keeps counting in Continuous mode.

This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Bit 1 **SNGSTRT:** LPTIM start in Single mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = 00), setting this bit starts the LPTIM in single pulse mode.

If the software start is disabled (TRIGEN[1:0] different than 00), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.

If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM stops at the following match between LPTIM_ARR and LPTIM_CNT registers.

This bit can only be set when the LPTIM is enabled. It is automatically reset by hardware.

Bit 0 **ENABLE:** LPTIM enable

The ENABLE bit is set and cleared by software.

0: LPTIM is disabled. Writing '0' to the ENABLE bit resets all the DMA request signals (input capture and update event DMA requests).

1: LPTIM is enabled

30.7.9 LPTIM compare register 1 (LPTIM_CCR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR1[15:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the capture/compare 1 register.

Depending on the PRELOAD option, the CCR1 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOAD bit is reset.

The capture/compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 becomes read-only, it contains the counter value transferred by the last input capture 1 event. The LPTIM_CCR1 register is read-only and cannot be programmed.

Caution: The LPTIM_CCR1 register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

30.7.10 LPTIM autoreload register (LPTIM_ARR)

Address offset: 0x018

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ARR[15:0]:** Auto reload value

ARR is the autoreload value for the LPTIM.

This value must be strictly greater than the CCRx[15:0] value.

Caution: The LPTIM_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

30.7.11 LPTIM counter register (LPTIM_CNT)

Address offset: 0x01C

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]:** Counter value

When the LPTIM is running, reading the LPTIM_CNT register may return unreliable values. In this case it is necessary to perform consecutive reads until two returned values are identical.

30.7.12 LPTIM configuration register 2 (LPTIM_CFGR2)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2SEL[1:0]		Res.	Res.	IC1SEL[1:0]	
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
										rw	rw			rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **IC2SEL[1:0]**: LPTIM input capture 2 selection

The IC2SEL bits control the LPTIM Input capture 2 multiplexer, which connects LPTIM Input capture 2 to one of the available inputs.
00: lptim_ic2_mux0
01: lptim_ic2_mux1
10: lptim_ic2_mux2
11: lptim_ic2_mux3
For connection details refer to *Section 30.4.3: LPTIM input and trigger mapping*.

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **IC1SEL[1:0]**: LPTIM input capture 1 selection

The IC1SEL bits control the LPTIM Input capture 1 multiplexer, which connects LPTIM Input capture 1 to one of the available inputs.
00: lptim_ic1_mux0
01: lptim_ic1_mux1
10: lptim_ic1_mux2
11: lptim_ic1_mux3
For connection details refer to *Section 30.4.3: LPTIM input and trigger mapping*.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **IN2SEL[1:0]**: LPTIM input 2 selection
The IN2SEL bits control the LPTIM input 2 multiplexer, which connects LPTIM input 2 to one of the available inputs.
00: lptim_in2_mux0
01: lptim_in2_mux1
10: lptim_in2_mux2
11: lptim_in2_mux3
For connection details refer to *Section 30.4.3: LPTIM input and trigger mapping*.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **IN1SEL[1:0]**: LPTIM input 1 selection
The IN1SEL bits control the LPTIM input 1 multiplexer, which connects LPTIM input 1 to one of the available inputs.
00: lptim_in1_mux0
01: lptim_in1_mux1
10: lptim_in1_mux2
11: lptim_in1_mux3
For connection details refer to *Section 30.4.3: LPTIM input and trigger mapping*.

30.7.13 LPTIM repetition register (LPTIM_RCR)

Address offset: 0x028

Reset value: 0x0000 0000

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition register value
REP is the repetition value for the LPTIM.

Caution: The LPTIM_RCR register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the auto reload match event, otherwise an unpredictable behavior may occur.

30.7.14 LPTIM capture/compare mode register 1 (LPTIM_CCMR1)

Address offset: 0x02C

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (PWM mode). The direction of a channel is defined by configuring the corresponding CCxSEL bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC2F[1:0]		Res.	Res.	IC2PSC[1:0]		Res.	Res.	Res.	Res.	CC2P[1:0]		CC2E	CC2 SEL
		rw	rw			rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	IC1F[1:0]		Res.	Res.	IC1PSC[1:0]		Res.	Res.	Res.	Res.	CC1P[1:0]		CC1E	CC1 SEL
		rw	rw			rw	rw					rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC2F[1:0]**: Input capture 2 filter

This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: Any external input capture signal level change is considered as a valid transition

01: External input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: External input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: External input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **IC2PSC[1:0]**: Input capture 2 prescaler

This bitfield defines the ratio of the prescaler acting on the CC2 input (IC2).

00: No prescaler, capture is done each time an edge is detected on the capture input

01: Capture is done once every 2 events

10: Capture is done once every 4 events

11: Capture is done once every 8 events

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:18 **CC2P[1:0]**: Capture/compare 2 output polarity.

Condition: CC2 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC2 active high

1: OC2 active low

Condition: CC2 as input

This field is used to select the IC2 polarity for capture operations.

00: Rising edge, circuit is sensitive to IC2 rising edge

01: Falling edge, circuit is sensitive to IC2 falling edge

10: Reserved, do not use this configuration.

11: Both edges, circuit is sensitive to both IC2 rising and falling edges.

Bit 17 **CC2E**: Capture/compare 2 output enable.

Condition: CC2 as output

0: Off - OC2 is not active. Writing '0' to the CC2E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC2 signal is output on the corresponding output pin

Condition: CC2 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 2 (LPTIM_CCR2) or not.

0: Capture disabled. Writing '0' to the CC2E bit resets the associated ic2_dma_req signal.

1: Capture enabled.

Bit 16 **CC2SEL**: Capture/compare 2 selection

This bitfield defines the direction of the channel, input (capture) or output mode.

0: CC2 channel is configured in output PWM mode

1: CC2 channel is configured in input capture mode

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **IC1F[1:0]**: Input capture 1 filter

This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: Any external input capture signal level change is considered as a valid transition

01: External input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: External input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: External input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (IC1).

00: No prescaler, capture is done each time an edge is detected on the capture input

01: Capture is done once every 2 events

10: Capture is done once every 4 events

11: Capture is done once every 8 events

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:2 **CC1P[1:0]**: Capture/compare 1 output polarity.

Condition: CC1 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC1 active high, the LPTIM output reflects the compare results between LPTIM_ARR and LPTIM_CCRx registers

1: OC1 active low, the LPTIM output reflects the inverse of the compare results between LPTIM_ARR and LPTIM_CCRx registers

Condition: CC1 as input

This field is used to select the IC1 polarity for capture operations.

00: Rising edge, circuit is sensitive to IC1 rising edge

01: Falling edge, circuit is sensitive to IC1 falling edge

10: Reserved, do not use this configuration.

11: Both edges, circuit is sensitive to both IC1 rising and falling edges.

Bit 1 **CC1E**: Capture/compare 1 output enable.

Condition: CC1 as output

0: Off - OC1 is not active. Writing '0' to the CC1E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC1 signal is output on the corresponding output pin

Condition: CC1 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (LPTIM_CCR1) or not.

0: Capture disabled. Writing '0' to the CC1E bit resets the associated ic1_dma_req signal.

1: Capture enabled.

Bit 0 **CC1SEL**: Capture/compare 1 selection

This bitfield defines the direction of the channel input (capture) or output mode.

0: CC1 channel is configured in output PWM mode

1: CC1 channel is configured in input capture mode

Caution: After a write to the LPTIM_CCMRx register, a new write operation to the same register can only be performed after a delay that must be equal or greater than the value of (PRESC × 3) kernel clock cycles, PRESC[2:0] being the clock decimal division factor (1, 2, 4,..128). Any successive write violating this delay, leads to unpredictable results.

Caution: The CCxSEL, ICxF[1:0], CCxP[1:0] and ICxPSC[1:0] fields must only be modified when the channel x is disabled (CCxE bit reset to 0).

30.7.15 LPTIM compare register 2 (LPTIM_CCR2)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR2[15:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the capture/compare 2 register.

Depending on the PRELOAD option, the CCR2 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOAD bit is reset.

The capture/compare register 2 contains the value to be compared to the counter LPTIM_CNT and signaled on OC2 output.

If channel CC2 is configured as input:

CCR2 becomes read-only, it contains the counter value transferred by the last input capture 2 event. The LPTIM_CCR2 register is read-only and cannot be programmed.

Caution: The LPTIM_CCR2 register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

Note: If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 30.3: LPTIM implementation*.

30.7.16 LPTIM register map

The following table summarizes the LPTIM registers.

Table 295. LPTIM register map and reset values

Offset	Register name	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
0x000	LPTIMx_ISR (x = 1, 2) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	CMP2OK[1]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IF[1]	REPOK	UE	DOWN[2]	UP[2]	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
	Reset value								0					0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_ISR (x = 1, 2) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF[1]	CC1OF	Res.	Res.	CC2IF[1]	REPOK	UE	DOWN[2]	UP[2]	ARROK	Res.	EXTTRIG	ARRM	CC1IF
	Reset value								0											0	0			0	0	0	0	0	0		0	0	0
0x004	LPTIMx_ICR (x = 1, 2) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	CMP2OKCF[1]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2CF[1]	REPOKCF	UECF	DOWNCF[2]	UPCF[2]	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0					0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_ICR (x = 1, 2) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OCF[1]	CC1OCF	Res.	Res.	CC2CF[1]	REPOKCF	UECF	DOWNCF[2]	UPCF[2]	ARROKCF	Res.	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0											0	0			0	0	0	0	0	0		0	0	0
0x008	LPTIMx_DIER (x = 1, 2) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UEDE	Res.	Res.	Res.	CMP2OKIE[1]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IE[1]	REPOKIE	UEIE	DOWNIE[2]	UPIE[2]	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
	Reset value									0				0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_DIER (x = 1, 2) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	CC2DE[1]	Res.	UEDE	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	Res.	Res.	CC2OIE[1]	CC1OIE	Res.	Res.	CC2IE[1]	REPOKIE	UEIE	DOWNIE[2]	UPIE[2]	ARROKIE	Res.	EXTTRIGIE	ARRMIE	CC1IE
	Reset value							0		0							0			0	0			0	0	0	0	0	0		0	0	0

Table 295. LPTIM register map and reset values (continued)

Offset	Register name	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
0x00C	LPTIM_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC(2)	COUNTMODE	PRELOAD	Res.	WAVE	TIMOUT	TRIGEN		Res.	TRIGSEL[2:0]			Res.	PRESC			Res.	TRGFLT		Res.	CKFLT		CKPOL		CKSEL
	Reset value								0	0	0		0	0	0	0		0	0	0		0	0	0		0	0		0	0	0	0	0
0x010	LPTIM_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
	Reset value																												0	0	0	0	0
0x014	LPTIM_CCR1	CCR1[15:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x018	LPTIM_ARR	ARR[15:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																1
0x01C	LPTIM_CNT	CNT[15:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x024	LPTIM_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2SEL[1:0]		Res.	Res.	IC1SEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
	Reset value											0	0			0	0											0	0			0	0
0x028	LPTIM_RCR	REP[7:0]																															
	Reset value																									0	0	0	0	0	0	0	0
0x02C	LPTIM_CCMR1	Res.	Res.	IC2F[1:0]		Res.	Res.	IC2PSC[1:0]		Res.	Res.	Res.	Res.	CC2P[1:0]		CC2E	CC2SEL	Res.	Res.	IC1F[1:0]		Res.	Res.	IC1PSC[1:0]		Res.	Res.	Res.	Res.	CC1P[1:0]		CC1E	CC1SEL
	Reset value			0	0			0	0					0	0	0	0			0	0			0	0					0	0	0	0
0x034	LPTIM_CCR2(3)	CCR2[15:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

1. If LPTIM does not implement at least 2 channels this bit is reserved. Refer to *Section 30.3: LPTIM implementation*.
2. If LPTIM does not support Encoder mode feature, this bit is reserved. Refer to *Section 30.3: LPTIM implementation*.
3. If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 30.3: LPTIM implementation*.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

31 Infrared interface (IRTIM)

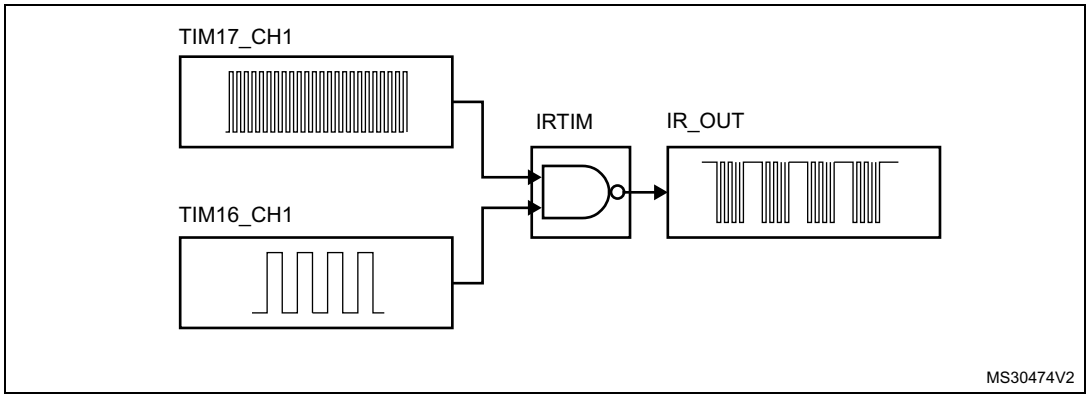
An infrared interface (IRTIM) for remote control is available on the device. It can be used with an infrared LED to perform remote control functions.

It uses internal connections with TIM16, and TIM17 as shown in *Figure 264*.

To generate the infrared remote control signals, the IR interface must be enabled and TIM16 channel 1 (TIM16_OC1) and TIM17 channel 1 (TIM17_OC1) must be properly configured to generate correct waveforms.

The infrared receiver can be implemented easily through a basic input capture mode.

Figure 264. IRTIM internal hardware connections with TIM16 and TIM17



All standard IR pulse modulation modes can be obtained by programming the two timer output compare channels.

TIM17 is used to generate the high frequency carrier signal, while TIM16 generates the modulation envelope.

The infrared function is output on the IR_OUT pin. The activation of this function is done through the GPIOx_AFRx register by enabling the related alternate function bit.

32 Independent watchdog (IWDG)

32.1 IWDG introduction

The independent watchdog (IWDG) peripheral offers a high safety level, thanks to its capability to detect malfunctions due to software or hardware failures.

The IWDG is clocked by an independent clock, and stays active even if the main clock fails.

In addition, the watchdog function is performed in the V_{DD} voltage domain, allowing the IWDG to remain functional even in low power modes. Refer to *Section 32.3* to check the capability of the IWDG in this product.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, making it very reliable to detect any unexpected behavior.

32.2 IWDG main features

- 12-bit down-counter
- Dual voltage domain, thus enabling operation in low power modes
- Independent clock
- Early wake-up interrupt generation
- Reset generation
  - In case of timeout
  - In case of refresh outside the expected window

32.3 IWDG implementation

Table 296. IWDG features (1)

IWDG modes/features	IWDG
LSI used as IWDG kernel clock (iwdg_ker_ck)	X
Window function	X
Early wake-up interrupt generation	X
System reset generation(2)	X
Capability to work in system Stop	X
Capability to work in system Standby	X
Capability to generate an interrupt in system Stop	X
Capability to generate an interrupt in system Standby	X
Capability to be frozen when the microcontroller enters in Debug mode(3)	X
Option bytes to control the activity in Stop mode(4)	X
Option bytes to control the activity in Standby mode(5)	X
Option bytes to control the Hardware mode(6)	X

1. ‘X’ = supported, ‘-’ = not supported.
2. Refer to the RCC section for additional information.
3. Controlled via DBG_IWDG_STOP in DBG section.

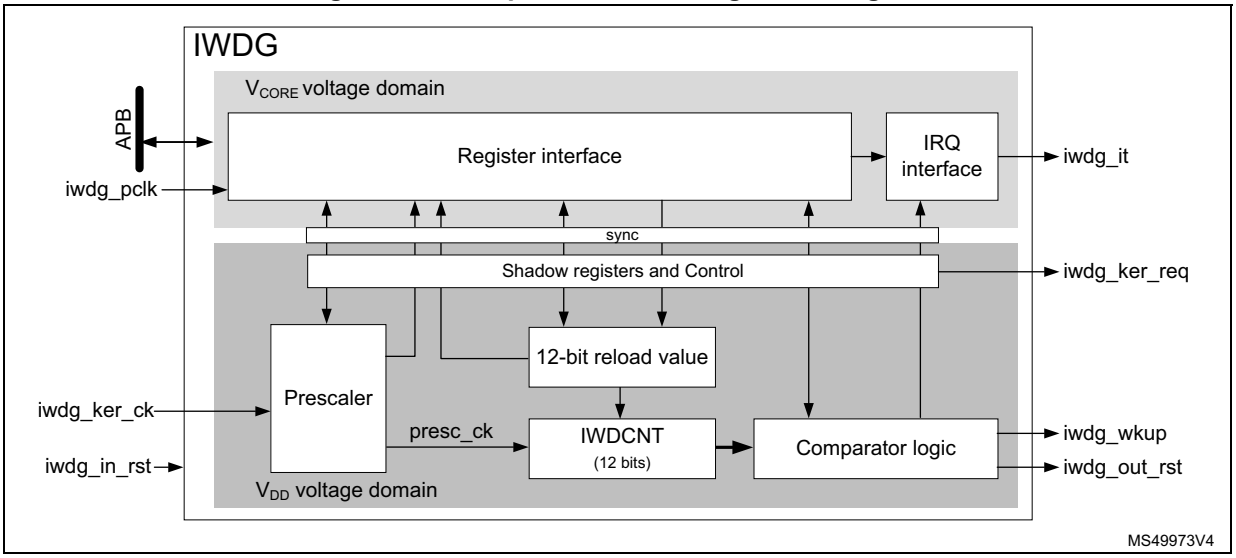
4. Controlled via the option byte IWDG_STOP in FLASH section.
5. Controlled via the option byte IWDG_STDBY in FLASH section.
6. Controlled via the option byte IWDG_SW in FLASH section.

32.4 IWDG functional description

32.4.1 IWDG block diagram

Figure 265 shows the functional blocks of the independent watchdog module.

Figure 265. Independent watchdog block diagram



The register and IRQ interfaces are located into the V_{CORE} voltage domain. The watchdog function itself is located into the V_{DD} voltage domain to remain functional in low power modes. See *Section 32.3* for IWDG capabilities.

The register and IRQ interfaces are mainly clocked by the APB clock (iwdg_pclk), while the watchdog function is clocked by a dedicated kernel clock (iwdg_ker_ck). A synchronization mechanism makes the data exchange between the two domains possible. Note that most of the registers located in the register interface are shadowed into the V_{DD} voltage domain.

The IWDG down-counter (IWDCNT) is clocked by the prescaled clock (presc_ck). The prescaled clock is generated from the kernel clock iwdg_ker_ck divided by the prescaler, according to PR[3:0] bitfield.

The table below gives the timing delays according to the actions performed on the IWDG: changing the prescaler, the timeout, the window or the early wake-up comparator values or doing a refresh.

Table 297. IWDG delays versus actions (1)

TD_{RVU}, TD_{PVU}		TD_{WVU}		TD_{EWU}		$TD_{Refresh}$		$TD_{RefAuto}$	
Min	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**
5 T_k	6 T_k	-	6 T_k	-	6 T_k	2 T_k	2 T_k + T_p	-	T_p

1. Tk represents a period of the kernel clock input, Tp represents a period of presc_ck.

32.4.2 IWDG internal signals

The list of IWDG internal signals is detailed in *Table 298*.

Table 298. IWDG internal input/output signals

Signal name	Signal type	Description
iwdg_ker_ck	Input	IWDG kernel clock
iwdg_ker_req	Input	IWDG kernel clock request
iwdg_pclk	Input	IWDG APB clock
iwdg_out_rst	Output	IWDG reset output
iwdg_in_rst	Input	IWDG reset input
iwdg_wkup	Output	IWDG wake-up event
iwdg_it	Output	IWDG early wake-up interrupt

32.4.3 Software and hardware watchdog modes

The watchdog modes allow the application to select the way the IWDG is enabled, either by software commands (Software watchdog mode), or automatically (Hardware watchdog mode). All other functions work similarly for both Software and Hardware modes.

The software watchdog mode is the default working mode. The independent watchdog is started by writing the value 0x0000 CCCC into the *IWDG key register (IWDG_KR)*, and the IWDCNT starts counting down from the reset value (0xFFF).

In the hardware watchdog mode the independent watchdog is started automatically at power-on, or every time it is reset (via iwdg_in_rst). The IWDCNT down-counter starts counting down from the reset value 0xFFF. The hardware watchdog mode feature is enabled through the device option bits, see *Section 32.3* for details.

When the IWDG is enabled the ONF flag is set to 1.

When the IWDCNT reaches 0x000, a reset signal is generated (iwdg_out_rst asserted).

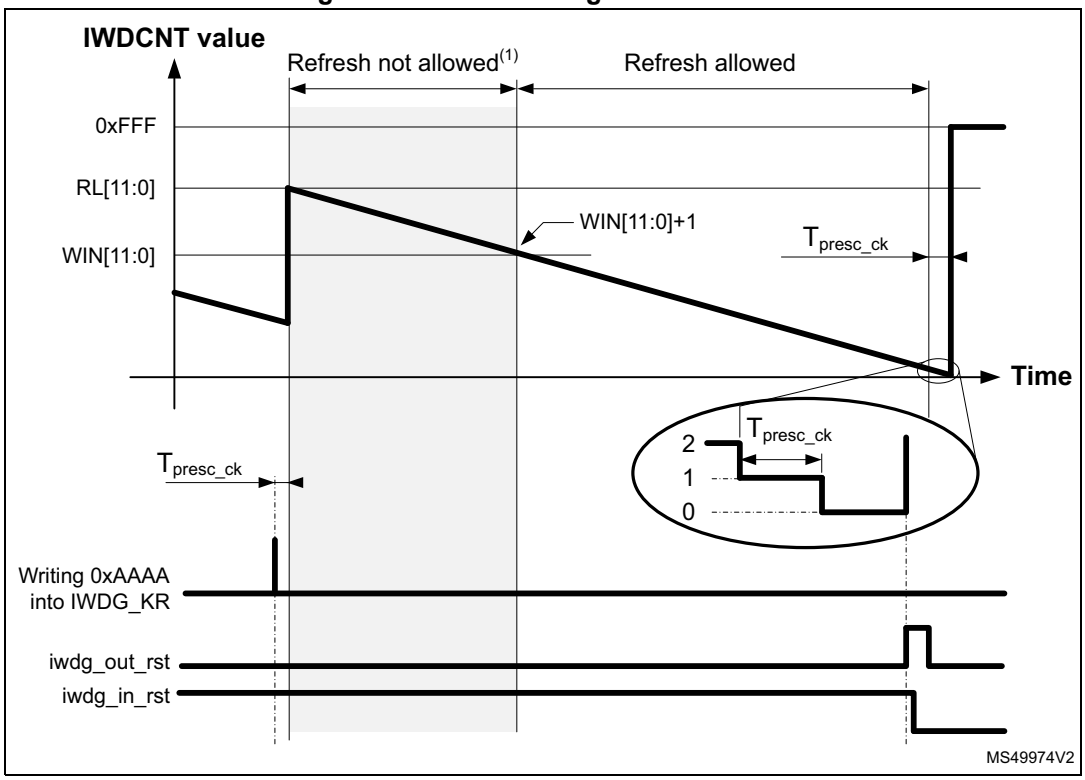
Whenever the key value 0x0000 AAAA is written in the *IWDG key register (IWDG_KR)*, the IWDG_RLR value is reloaded into the IWDCNT, and the watchdog reset is prevented.

Due to re-synchronization delays, the IWDG must be refreshed before the IWDCNT down-counter reaches 1.

Once started, the IWDG can be stopped only when it is reset (iwdg_in_rst asserted).

As shown in *Figure 266*, when the refresh command is executed, one period of presc_ck later, the IWDCNT is reloaded with the content of RL[11:0].

Figure 266. Reset timing due to timeout



1. If window option activated.

If the IWDG is not refreshed before the IWDCNT reaches 1, the IWDG generates a reset (iwdg_out_rst is asserted). In return, the RCC resets the IWDG (assertion of iwdg_in_rst) to clear the reset source.

32.4.4 Window option

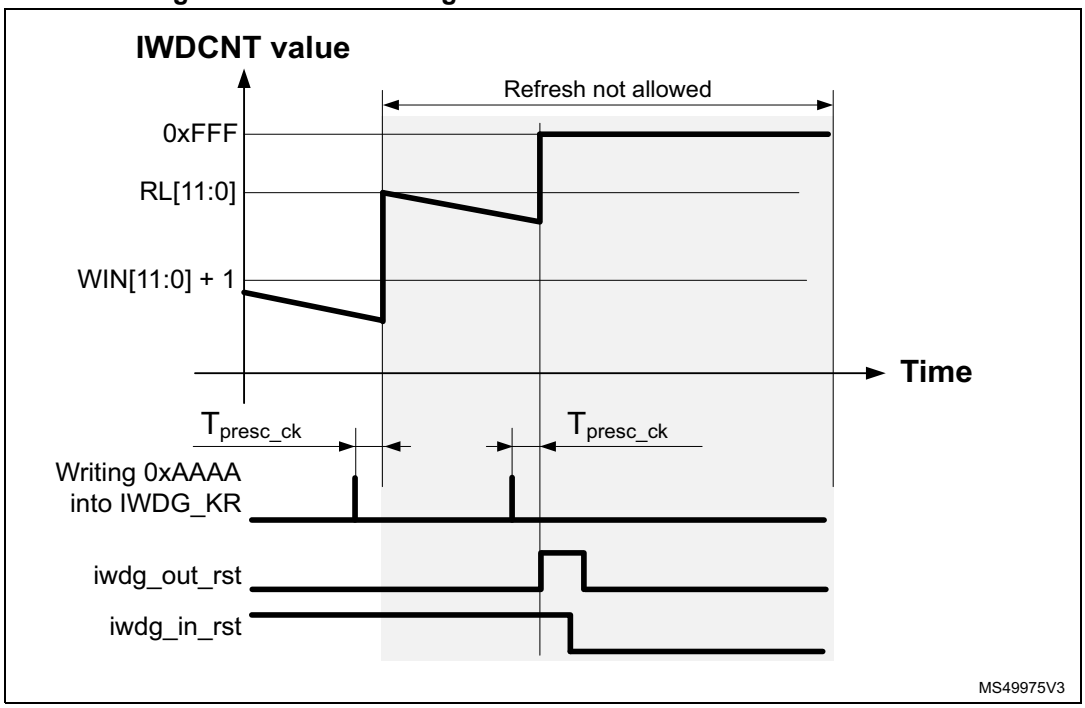
The IWDG can also work as a window watchdog, by setting the appropriate window in the *IWDG window register (IWDG_WINR)*.

If the reload operation is performed while the counter is greater than WIN[11:0] + 1, a reset is generated. WIN[11:0] is located in the *IWDG window register (IWDG_WINR)*. As shown in *Figure 267*, the reset is generated one period of presc_ck after the unexpected refresh command.

The default value of the *IWDG window register (IWDG_WINR)* is 0x0000 0FFF, so, if not updated, the window option is disabled.

As soon as the window value changes, the down-counter (IWDCNT) is reloaded with the RL[11:0] value, to ease the estimation for where the next refresh must take place.

Figure 267. Reset timing due to refresh in the not allowed area



Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the IWDG prescaler by programming *IWDG prescaler register (IWDG_PR)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Write to the *IWDG window register (IWDG_WINR)*. This automatically reloads the IWDCNT down-counter with the RL[11:0] value.
7. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
8. Write 0x0000 0000 into *IWDG key register (IWDG_KR)* to write-protect registers.

Note: Step 7 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

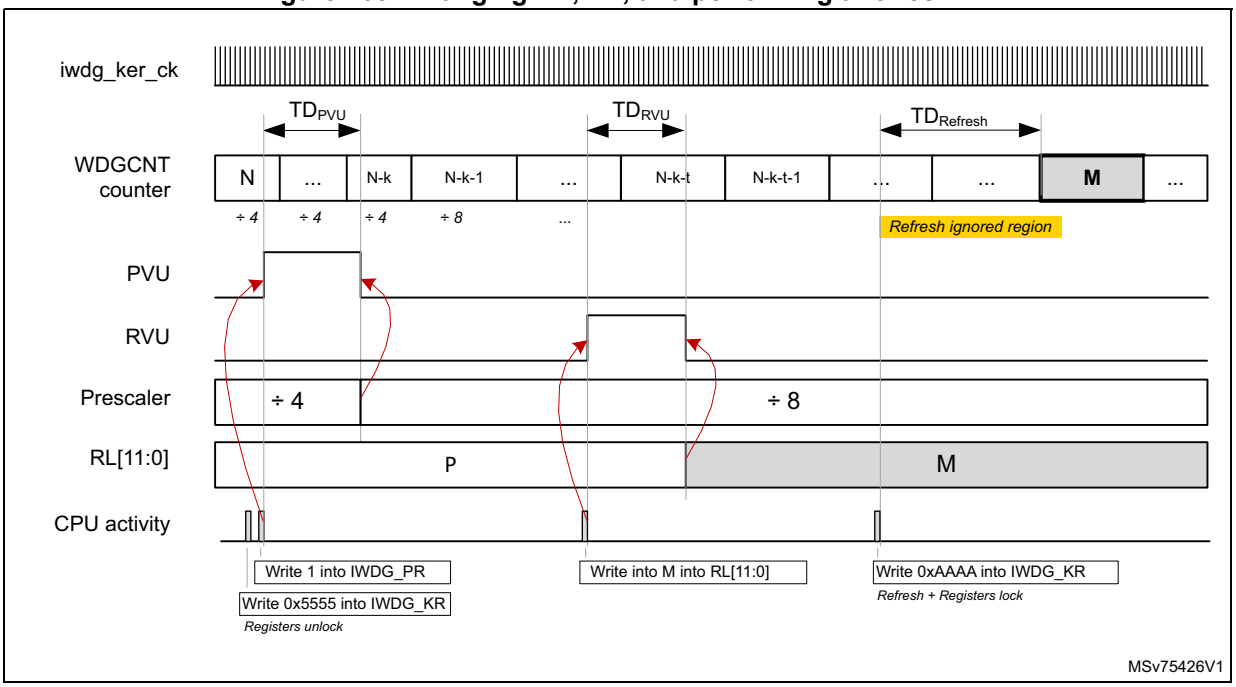
Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the prescaler by programming the *IWDG prescaler register (IWDG_PR)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
7. Refresh the counter with RL[11:0] value, and write-protect registers by writing 0x0000 AAAA into *IWDG key register (IWDG_KR)*.

The figure below shows a sequence example changing the prescaler, the reload value, and then performing a refresh.

Figure 268. Changing PR, RL, and performing a refresh(1)



1. Refer to *Table 297: IWDG delays versus actions* for details on timing values.

Note: When the new prescaler value is accepted by the IWDG (falling edge of PVU), the new division ratio is effective when the current division sequence is completed (N-k in the drawing).

Note: The timeout delay between the refresh (write 0xAAAA into IWDG_KR) and the watchdog reset is increased by TDRefresh.

If a refresh command is sent while the previous refresh command is not yet completed, this last refresh command is ignored.

Updating the window comparator

It is possible to update the window comparator when the IWDG is already running. The IWDCNT is reloaded as well. The following sequence can be performed to update the window comparator:

1. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
2. Write to the IWDG window register (IWDG_WINR). This automatically reloads the IWDCNT down-counter with RL[11:0] value.
3. Wait for WVU = 0
4. Lock registers by writing IWDG_KR to 0x0000 0000

Step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

32.4.5 Debug

When the processor enters into Debug mode (core halted), the IWDCNT down-counter either continues to work normally or stops, depending on debug capability of the product. Refer to *Section 32.3* for details on the capabilities of this product.

32.4.6 Register access protection

Write accesses to *IWDG prescaler register (IWDG_PR)*, *IWDG reload register (IWDG_RLR)*, *IWDG early wake-up interrupt register (IWDG_EWCR)* and *IWDG window register (IWDG_WINR)* are protected. To modify them, first write 0x0000 5555 in the *IWDG key register (IWDG_KR)*. A write access to this register with a different value breaks the sequence and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value or the window value is ongoing.

32.5 IWDG low power modes

Depending on option bytes configuration, the IWDG can continue counting or not during the low power modes. Refer to *Section 32.3* for details.

Table 299. Effect of low power modes on IWDG

Mode	Description
Sleep	No effect. IWDG interrupts cause the device to exit from the mode.
Stop	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 32.3* for details. IWDG interrupts cause the device exit the Stop mode.
Standby	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 32.3* for details. IWDG interrupts cause the device to exit from Standby mode.

32.6 IWDG interrupts

The IWDG offers the possibility to generate an early interrupt depending on the value of the down-counter. The early interrupt is enabled by setting the EWIE bit of the *IWDG early wake-up interrupt register (IWDG_EWCR)* to 1.

A comparator value (EWIT[11:0]) allows the application to define the position where the early interrupt must be generated.

When the IWDCNT down-counter reaches the value of EWIT[11:0] - 1, the iwdg_wkup is activated, making it possible for the system to exit from low power modes, if needed.

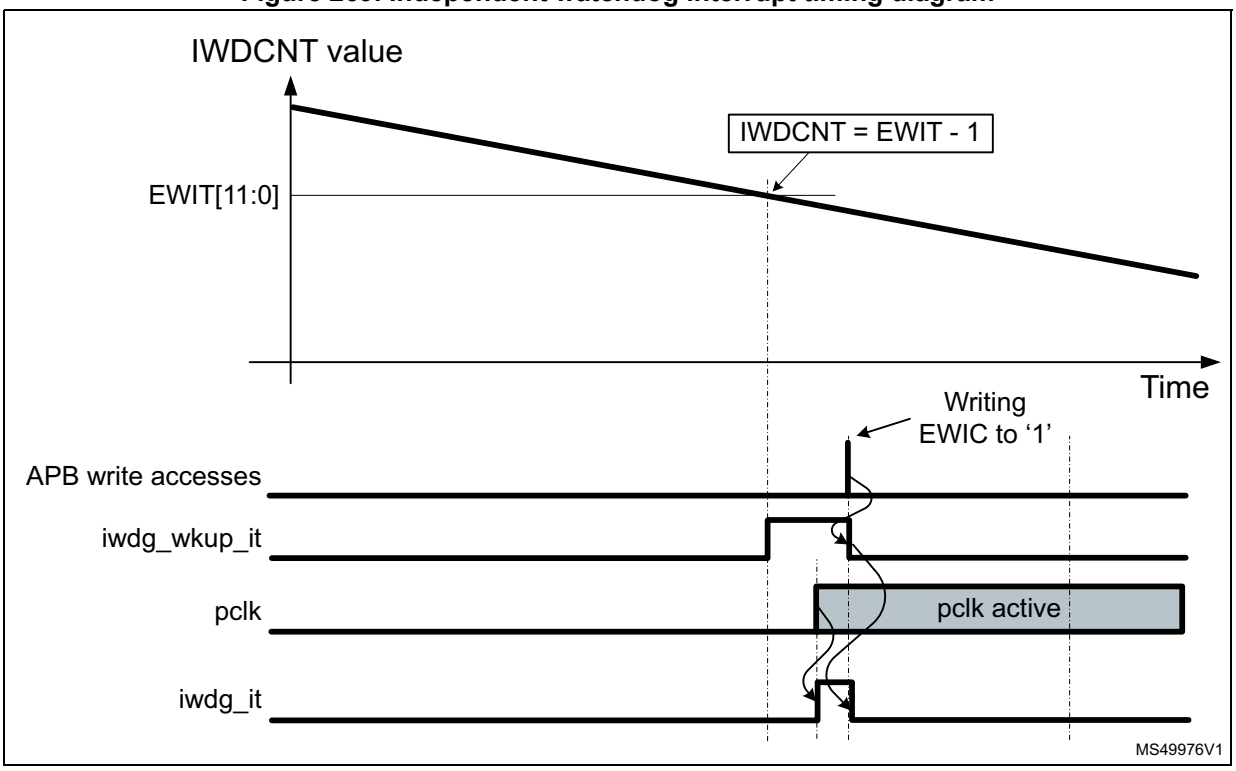
When the APB clock is available, the iwdg_it is activated as well.

In addition, the flag EWIF of the *IWDG status register (IWDG_SR)* is set to 1.

The EWI interrupt is acknowledged by writing 1 to the EWIC bit in the *IWDG early wake-up interrupt register (IWDG_EWCR)*.

Writing into the IWDG_EWCR register also triggers a refresh of the down-counter (IWDCNT) with the reload value RL[11:0].

Figure 269. Independent watchdog interrupt timing diagram



The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the watchdog reset is generated.

Changing the early wake-up comparator value

It is possible to change the early wake-up comparator value or to enable/disable the interrupt generation at any time, by performing the following sequence:

1. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.

2. Enable or disable the early wake-up interrupt, and/or program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
3. Wait for EWU = 0, EWU is located into the *IWDG status register (IWDG_SR)*.
4. Write-protect registers by writing 0x0000 0000 to *IWDG key register (IWDG_KR)*.

Step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Table 300 summarizes the IWDG interrupt request.

Table 300. IWDG interrupt request

Interrupt event	Event flag	Interrupt clear method	Interrupt enable control bit	Activated interrupt	
				iwdg_it	iwdg_wkup_it
IWDCNT reaches EWIT value	EWIF	Writing EWIC to 1	EWIE	Y(1)	Y(2)

1. Generated when a clock is present on iwdg_pclk input.
2. Generated when a clock is present on iwdg_ker_ck input.

32.7 IWDG registers

Refer to *Section 1.2 on page 66* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

Most of the registers located into the register interface are shadowed into the V_{DD} voltage domain. When the iwdg_in_rst is asserted, the watchdog logic and the shadow registers located into the V_{DD} voltage domain are reset.

When the application reads back a watchdog register, the hardware transfers the value of the corresponding shadow register to the register interface.

When the application writes a watchdog register, the hardware updates the corresponding shadow register.

32.7.1 IWDG key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0x0000)

These bits can be used for several functions, depending upon the value written by the application:

- 0xAAAA: reloads the RL[11:0] value into the IWDCNT down-counter (watchdog refresh), and write-protects registers. This value must be written by software at regular intervals, otherwise the watchdog generates a reset when the counter reaches 0.
- 0x5555: enables write-accesses to the registers.
- 0xCCCC: enables the watchdog (except if the hardware watchdog option is selected) and write-protects registers.
- values different from 0x5555: write-protects registers.

Note that only IWDG_PR, IWDG_RLR, IWDG_EWCR and IWDG_WINR registers have a write-protection mechanism.

32.7.2 IWDG prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PR[3:0]:** Prescaler divider

These bits are write access protected, see *Section 32.4.6*. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the *IWDG status register (IWDG_SR)* must be reset to be able to change the prescaler divider.

0000: divider / 4
0001: divider / 8
0010: divider / 16
0011: divider / 32
0100: divider / 64
0101: divider / 128
0110: divider / 256
0111: divider / 512
Others: divider / 1024

Note: Reading this register returns the prescaler value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG status register (IWDG_SR) is reset.

32.7.3 IWDG reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RL[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected, see *Section 32.4.6*. They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the *IWDG key register (IWDG_KR)*. The watchdog counter counts down from this value. The timeout period is a function of this value and the prescaler.clock. It is not recommended to set RL[11:0] to a value lower than 2.

The RVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the RVU bit in the IWDG status register (IWDG_SR) is reset.

32.7.4 IWDG status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (0xFFFF FEFF)

This register contains various status flags. Note that the mask value between parenthesis means that the reset value of ONF bit is not defined. When the IWDG is configured in software mode, the reset value of ONF bit is 0, when the IWDG is configured in hardware mode, the reset value of ONF bit is 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EWIF	Res.	Res.	Res.	Res.	Res.	Res.	ONF	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
r							r					r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIF**: Watchdog early interrupt flag

This bit is set to ‘1’ by hardware in order to indicate that an early interrupt is pending. This bit must be cleared by the software by writing the bit EWIC of IWDG_EWCR register to ‘1’.

Bits 14:9 Reserved, must be kept at reset value.

Bit 8 **ONF**: Watchdog enable status bit

Set to ‘1’ by hardware as soon as the IWDG is started. In software mode, it remains to '1' until the IWDG is reset. In hardware mode, this bit is always set to '1'.

0: The IWDG is not activated

1: The IWDG is activated and needs to be refreshed regularly by the application

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **EWU**: Watchdog interrupt comparator value update

This bit is set by hardware to indicate that an update of the interrupt comparator value (EWIT[11:0]) or an update of the EWIE is ongoing. It is reset by hardware when the update operation is completed in the V_{DD} voltage domain. Refer to *Table 297: IWDG delays versus actions* for delay values.

The EWIT[11:0] and EWIE fields can be updated only when EWU bit is reset.

Bit 2 **WVU:** Watchdog counter window value update

This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain. Refer to *Table 297: IWDG delays versus actions* for delay values.

The window value can be updated only when WVU bit is reset.

This bit is generated only if generic “window” = 1.

Bit 1 **RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain. Refer to *Table 297: IWDG delays versus actions* for delay values.

The reload value can be updated only when RVU bit is reset.

Bit 0 **PVU:** Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain. Refer to *Table 297: IWDG delays versus actions* for delay values.

The prescaler value can be updated only when PVU bit is reset.

Note: If several reload, prescaler, early interrupt position or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, to wait until WVU bit is reset before changing the window value, and to wait until EWU bit is reset before changing the early interrupt position value. After updating the prescaler and/or the reload/window/early interrupt value, it is not necessary to wait until RVU or PVU or WVU or EWU is reset before continuing code execution, except in case of low power mode entry.

32.7.5 IWDG window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **WIN[11:0]:** Watchdog counter window value

These bits are write access protected, see *Section 32.4.6*.They contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the IWDCNT downcounter must be reloaded when its value is lower than WIN[11:0] + 1 and greater than 1.

The WVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the IWDG status register (IWDG_SR) is reset.

32.7.6 IWDG early wake-up interrupt register (IWDG_EWCR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EWIE	EWIC	Res.	Res.	EWIT[11:0]											
rw	w			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIE**: Watchdog early interrupt enable

Set and reset by software.

0: The early interrupt interface is disabled.

1: The early interrupt interface is enabled.

The EWU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the value of this bit.

Bit 14 **EWIC**: Watchdog early interrupt acknowledge

The software must write a 1 into this bit in order to acknowledge the early wake-up interrupt and to clear the EWIF flag. Writing 0 has not effect, reading this flag returns a 0.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:0 **EWIT[11:0]:** Watchdog counter early wake-up interrupt value

These bits are write access protected (see *Section 32.4.6*). They are written by software to define at which position of the IWDCNT down-counter the early wake-up interrupt must be generated. The early interrupt is generated when the IWDCNT is lower or equal to EWIT[11:0] - 1.

EWIT[11:0] must be bigger than 1.

An interrupt is generated only if EWIE = 1.

The EWU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the Early wake-up comparator value and the Interrupt enable bit from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the EWU bit in the IWDG status register (IWDG_SR) is reset.

32.7.7 IWDG register map

Table 301. IWDG register map and reset values

Offset	Register name	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
0x00	IWDG_KR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	IWDG_PR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[3:0]			
	Reset value																													0	0	0	0
0x08	IWDG_RLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RL[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x0C	IWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF		Res.	Res.	Res.	Res.	Res.	ONF	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
	Reset value																	0							x					0	0	0	0
0x10	IWDG_WINR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WIN[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x14	IWDG_EWCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIE	EWIC	Res.	Res.	EWIT[11:0]											
	Reset value																	0	0			0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

33 Real-time clock (RTC)

33.1 RTC introduction

The RTC provides an automatic wake-up to manage all low-power modes, except Stop 3 and Standby.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode except Stop 3 and Standby, or under reset).

33.2 RTC main features

The RTC supports the following features (see *Figure 270: RTC block diagram*):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
- Binary mode with 32-bit free-running counter.
- Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
- Two programmable alarms.
- On-the-fly correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
- Timestamp feature which can be used to save the calendar content. This function can be triggered by an event on the timestamp pin, or by a tamper event.
- 17-bit auto-reload wake-up timer (WUT) for periodic events with programmable resolution and period.
- TrustZone support:
  - RTC fully securable
  - Alarm A, alarm B, wake-up Timer and timestamp individual secure or nonsecure configuration
- Alarm A, alarm B, wake-up Timer and timestamp individual privilege protection

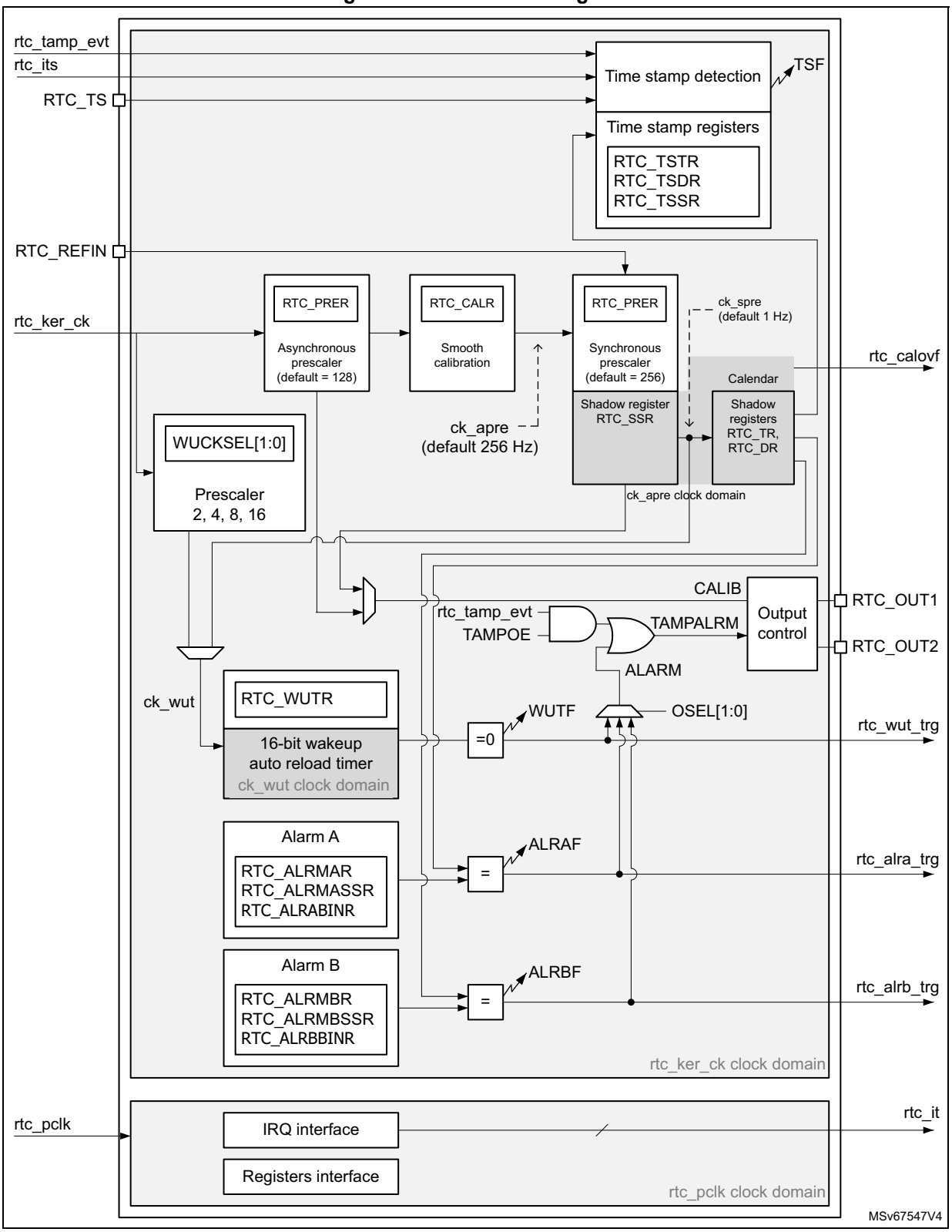
The RTC is functional in all low-power modes except Stop 3 and Standby when it is clocked by the LSE.

All RTC events (Alarm, wake-up Timer, Timestamp) can generate an interrupt and wake-up the device from the low-power modes.

33.3 RTC functional description

33.3.1 RTC block diagram

Figure 270. RTC block diagram



33.3.2 RTC pins and internal signals

Table 302. RTC input/output pins

Pin name	Signal type	Description
RTC_TS	Input	RTC timestamp input
RTC_REFIN	Input	RTC 50 or 60 Hz reference clock input
RTC_OUT1	Output	RTC output 1
RTC_OUT2	Output	RTC output 2

RTC_OUT1 and RTC_OUT2 which select one of the following two outputs:

- CALIB: 512 Hz or 1 Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC_CR register.
- TAMPALRM: This output is the OR between rtc_tamp_evt and ALARM signals.

ALARM is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the alarm A, alarm B or wake-up outputs. rtc_tamp_evt is enabled by setting the TAMPOE bit in the RTC_CR register which selects the tamper event outputs.

Table 303. RTC internal input/output signals

Internal signal name	Signal type	Description
rtc_ker_ck	Input	RTC kernel clock, also named RTCCLK in this document
rtc_pclk	Input	RTC APB clock
rtc_tamp_evt	Input	Tamper event (internal or external) detected in TAMP peripheral
rtc_tzen	Input	RTC TrustZone enabled
rtc_it	Output	RTC interrupts (refer to *Section 33.5: RTC interrupts* for details)
rtc_alra_trg	Output	RTC alarm A event detection trigger
rtc_alrb_trg	Output	RTC alarm B event detection trigger
rtc_wut_trg	Output	RTC wake-up timer event detection trigger
rtc_calovf	Output	RTC calendar overflow: this signal is generated when the RTC calendar reaches its maximum value, on the 31st of December 99, at 23:59:59. The calendar is then frozen and cannot overflow.

The RTC kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some functions are not available in some low-power modes when the selected clock is not LSE. Refer to *Section 33.4: RTC low-power modes* for more details.

Table 304. RTC interconnection

Signal name	Source/destination
rtc_tamp_evt	From TAMP peripheral: tamp_evt
rtc_calovf	To TAMP peripheral: tamp_itamp5

The triggers outputs can be used as triggers for other peripherals.

33.3.3 GPIOs controlled by the RTC and TAMP

The GPIOs included in the backup domain (V_{DD}) are directly controlled by the peripherals providing functions on these I/Os, whatever the GPIO configuration.

Both RTC and TAMP peripherals provide functions on these I/Os (refer to *Section 34: Tamper and backup registers (TAMP)*).

RTC_OUT1, RTC_TS, TAMP_IN4 and TAMP_OUT3 are mapped on the same pin (PC13). The RTC and TAMP functions mapped on PC13 are available in all low-power modes except Standby.

The output mechanism follows the priority order shown in *Table 305: RTC pin PC13 configuration*.

Table 305. RTC pin PC13 configuration(1)

PC13 Pin function	OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP3E=TAMP3AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=1	TAMP4E (TAMP_IN4 input enable)	TSE (RTC_TS input enable)
TAMPALRM output Push-Pull	01 or 10 or 11	0	Don't care	Don't care	0	0	Don't care	Don't care	Don't care
	00	1							
	01 or 10 or 11	1							

Table 305. RTC pin PC13 configuration(1) (continued)

PC13 Pin function		OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP3E=TAMP3AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=1	TAMP4E (TAMP_IN4 input enable)	TSE (RTC_TS input enable)
TAMPALRM output Open-Drain(2)	No pull	01 or 10 or 11	0	Don't care	Don't care	1	0	Don't care	Don't care	Don't care
		00	1							
		01 or 10 or 11	1							
	Internal pull-up	01 or 10 or 11	0	Don't care	Don't care	1	1	Don't care	Don't care	Don't care
		00	1							
		01 or 10 or 11	1							
CALIB output PP		00	0	1	0	Don't care	Don't care	Don't care	Don't care	Don't care
TAMP_OUT3 output PP		00	0	0	0	Don't care	Don't care	1	Don't care	Don't care
TAMP_IN4 input floating		00	0	0	Don't care	Don't care	Don't care	0	1	0
		00	0	1	1					
		Don't care	Don't care	0						
RTC_TS and TAMP_IN4 input floating		00	0	0	Don't care	Don't care	Don't care	0	1	1
		00	0	1	1			0		
		Don't care	Don't care	0				0		
RTC_TS input floating		00	0	0	Don't care	Don't care	Don't care	0	0	1
		00	0	1	1			0		
		Don't care	Don't care	0				0		

Table 305. RTC pin PC13 configuration(1) (continued)

PC13 Pin function	OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP3E=TAMP3AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=1	TAMP4E (TAMP_IN4 input enable)	TSE (RTC_TS input enable)
Wake-up pin or Standard GPIO	00	0	0	Don't care	Don't care	Don't care	0	0	0
	00	0	1	1			0		
	Don't care	Don't care	0				0		

1. OD: open drain; PP: push-pull.
2. In this configuration the GPIO must be configured in input.

In addition, it is possible to output RTC_OUT2 on PB2 pin thanks to OUT2EN bit. The different functions are mapped on RTC_OUT1 or on RTC_OUT2 depending on OSEL, COE and OUT2EN configuration, as shown in *Table 306: RTC_OUT mapping*.

Table 306. RTC_OUT mapping

RTC_OUT mapping				
OSEL[1:0] bits ALARM output enable)	COE bit (CALIB output enable)	OUT2EN bit	RTC_OUT1 on PC13	RTC_OUT2 on PB2
00	0	0	-	-
00	1		CALIB	-
01 or 10 or 11	Don't care		TAMPALRM	-
00	0	1	-	-
00	1		-	CALIB
01 or 10 or 11	0		-	TAMPALRM
01 or 10 or 11	1		TAMPALRM	CALIB

33.3.4 RTC secure protection modes

By default after a backup domain power-on reset, all RTC registers can be read or written in both secure and nonsecure modes, except for the RTC secure configuration register

(RTC_SECCFGR) which can be written in secure mode only. The RTC protection configuration is not affected by a system reset.

When the SEC bit is set in the RTC_SECCFGR register:

- Writing the RTC registers is possible only in secure mode.
- Reading RTC_SECCFGR, RTC_PRIVCFGR, RTC_MISR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in secure and nonsecure modes. All the other RTC registers can be read only in secure mode.

When the SEC bit is cleared, it is still possible to protect some of the registers by setting dedicated INITSEC, CALSEC, TSSEC, WUTSEC, ALRASEC or ALRBSEC control bits. If all these bits are also clear, all the RTC registers can be read and written in secure and nonsecure mode.

- When INITSEC is set:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT, BIN and BCDU in RTC_ICSR, FMT control bits in RTC_CR and INITPRIV in the RTC_PRIVCFGR can be written only in secure mode.
  - These registers and control bits can be read in secure and nonsecure mode.
- When CALSEC is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR and CALPRIV in the RTC_PRIVCFGR can be written only in secure mode.
  - These registers and control bits can be read in secure and nonsecure mode.
- When ALRASEC is set:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE, ALRAFCLR, ALRAIE and SSRUIE in the RTC_CR, CALRAF and CSSRUF in the RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_SMISR can be read and written only in secure mode.
  - ALRAPRIV in the RTC_PRIVCFGR can be written only in secure mode
- When ALRBSEC is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE ALRBFCLR, ALRBIE in the RTC_CR, CALRBF in the RTC_SCR, ALRBF in RTC_SR, and ALRBMF in RTC_SMISR can be read and written only in secure mode.
  - ALRBPRIV in the RTC_PRIVCFGR can be written only in secure mode.
- When WUTSEC is set:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_SMISR can be read and written only in secure mode.
  - WUTPRIV in the RTC_PRIVCFGR can be written only in secure mode
- When TSSEC is set:
  - RTC_TSTR, RTC_TSDR and RTC_TSSSR registers, plus TAMPTS, TSE, TSIE, TSEDGE control bits in the RTC_CR, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF in RTC_SR, and TSMF, TSOVMF in RTC_SMISR can be read and written only in secure mode.
  - TSPRIV in the RTC_PRIVCFGR can be written only in secure mode

A nonsecure access to a secure-protected register is denied:

- There is no bus error generated.
- In case the register has a global protection, excepting RTC_WPR: a notification is generated through a flag/interrupt in the TZIC (TrustZone illegal access controller). In case only a few bits of the register are protected (for registers with mixed features such as RTC_CR...), no notification is generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

As soon as at least one function is configured to be secured, the RTC reset and clock control is also secured in the RCC.

33.3.5 RTC privilege protection modes

By default after a backup domain power-on reset, all RTC registers can be read or written in both privileged and unprivileged modes, except for the RTC privilege mode control register (RTC_PRIVCFGR) which can be written in privilege mode only. The RTC protection configuration is not affected by a system reset.

When the PRIV bit is set in the RTC_PRIVCFGR register:

- Writing the RTC registers is possible only in privileged mode.
- Reading the RTC_SECCFGR, RTC_PRIVCFGR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in privilege and unprivilege modes. All the other RTC registers can be read only in privilege mode.

When the PRIV bit is cleared, it is still possible to protect some of the registers by setting dedicated INITPRIV, CALPRIV, TSPRIV, WUTPRIV, ALRAPRV or ALRBPRIV control bits. If all these bits are also cleared, all the RTC registers can be read or written in privilege and unprivilege modes.

- When INITPRIV is set:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT, BIN and BCDU in RTC_ICSR and FMT control bits in RTC_CR, plus INITSEC in the RTC_SECCFGR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and unprivilege mode.
- When CALPRIV is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR, plus CALDSEC in the RTC_SECCFGR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and unprivilege mode.
- When ALRAPRIV is set:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE, ALRAFCLR, ALRAIE and SSRUIE in the RTC_CR, and CALRAF and CSSRUF in the RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - ALRASEC in the RTC_SECCFGR can be written only in privilege mode.
- When ALRBPRIV is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE, ALRBFCLR, ALRBIE in the RTC_CR, and CALRBF in the RTC_SCR, ALRBF in

RTC_SR, and ALRBMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.

  - ALRBSEC in the RTC_SECCFGR can be written only in privilege mode.
- When WUTPRIV is set:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, and CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - WUTSEC in the RTC_SECCFGR can be written only in privilege mode.
- When TSPRIV is set:
  - RTC_TSTR, RTC_TSDR and RTC_TSSSR registers, plus TAMPTS, TSE, TSIE, TSEDGE control bits in the RTC_CR, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF in RTC_SR, and TSMF, TSOVMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - TSSEC in the RTC_SECCFGR can be written only in privilege mode.

A unprivileged access to a privileged-protected register is denied:

- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

33.3.6 Clock and prescalers

The RTC clocks must respect this ratio: frequency(PCLK) ≥ 2 × frequency(RTCCLK).

For more information on the RTC clock (RTCCLK) source configuration, refer to “Reset and clock control (RCC)”.

BCD mode (BIN=00)

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see *Figure 270: RTC block diagram*):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

Note: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is 2^{22}.

This corresponds to a maximum input frequency of around 4 MHz.

f_{ck_apre} is given by the following formula:

$$f_{CK_APRE} = \frac{f_{RTCCLK}}{PREDIV_A + 1}$$

The ck_apre clock is used to clock the binary RTC_SSR subsecond downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

f_{ck_spre} is given by the following formula:

$$f_{CK_SPRE} = \frac{f_{RTCCLK}}{(PREDIV_S + 1) \times (PREDIV_A + 1)}$$

The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wake-up auto-reload timer. To obtain short timeout periods, the 16-bit wake-up auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see *Section 33.3.10: Periodic auto-wake-up* for details).

Binary mode (BIN=01)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are not functional.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.

PREDIV_S value is don't care.

Mixed mode (BIN=10 or 11)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are also available.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register. The bits BCDU[2:0] are used to define when the calendar is incremented by 1 second, using the SSR least significant bits.

33.3.7 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ICSR register is set (see *Section 33.6.12: RTC shift control register (RTC_SHIFTR)*). The copy is not performed in Stop 0, 1, 2, and Standby retention mode. When exiting these modes, the shadow registers are updated after up to 4 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD = 0 mode, the frequency of the APB clock (f_{APB}) must be at least 7 times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

33.3.8 Calendar ultra-low power mode

It is possible to reduce drastically the RTC power consumption by setting the LPCAL bit in the RTC_CALR register. In this configuration, the whole RTC is clocked by ck_apre only instead of both RTCCLK and ck_apre. Consequently, some flags delays are longer, and the calibration window is longer (refer to *Section : RTC ultra-low-power mode*).

The LPCAL bit is ignored (assumed to be 0) when asynchronous prescaler division factor (PREDIV_A+1) is not a power of 2.

Switching from LPCAL=0 to LPCAL=1 or from LPCAL=1 to LPCAL=0 is not immediate and requires a few ck_apre periods to complete.

33.3.9 Programmable alarms

The RTC unit provides programmable alarm: alarm A and alarm B. The description below is given for alarm A, but can be translated in the same way for alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC_CR register.

The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASSR register.

When the binary mode is used, the subsecond field can be programmed in the alarm binary register RTC_ALRABINR.

The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

In case the Alarm is used to generate a trigger event for another peripheral, the ALRAF can be automatically cleared by hardware by configuring the ALRAFCLR bit at 1 in the RTC_CR register. In this configuration there is no need for software intervention if the only purpose is clearing the ALRAF flag.

Caution: If the seconds field is selected (MSK1 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

Alarm A and alarm B (if enabled by bits OSEL[1:0] in RTC_CR register) can be routed to the TAMPALRM output. TAMPALRM output polarity can be configured through bit POL the RTC_CR register.

33.3.10 Periodic auto-wake-up

The periodic wake-up flag is generated by a 16-bit programmable auto-reload down-counter. The wake-up timer range can be extended to 17 bits.

The wake-up function is enabled through the WUTE bit in the RTC_CR register.

The wake-up timer clock input ck_wut can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.

  When RTCCLK is LSE (32.768 kHz), this permits the wake-up interrupt period to be configured from 122 µs to 32 s, with a resolution down to 61 µs.
- ck_spre (usually 1 Hz internal clock) in BCD mode, or the clock used to update the calendar as defined by BCDU in binary or mixed (BCD-binary) modes.

  When ck_spre frequency is 1 Hz, a wake-up time from 1 s to around 36 hours can be achieved with one-second resolution. This large programmable time range is divided in 2 parts:
  - from 1 s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18 h to 36 h when WUCKSEL[2:1] = 11. In this last case 2^{16} is added to the 16-bit counter current value. When the initialization sequence is complete (see *Programming the wake-up timer on page 1116*), the timer starts counting down. When the wake-up function is enabled, the down-counting remains active in low-power modes except Stop 3 and Standby. In addition, when it reaches 0, the WUTF flag is set in the RTC_SR register, and the wake-up counter is automatically reloaded with its reload value (RTC_WUTR register value).

Depending on WUTOCLR in the RTC_WUTR register, the WUTF flag must either be cleared by software (WUTOCLR = 0x0000), or the WUTF is automatically cleared by hardware when the auto-reload down counter reaches WUTOCLR value (0x0000 < WUTOCLR ≤ WUT).

The wake-up flag is output on an internal signal rtc_wut that can be used by other peripherals (refer to section *Section 33.3.1: RTC block diagram*).

When the periodic wake-up interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes except Stop 3 and Standby.

The periodic wake-up flag can be routed to the TAMPALRM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. TAMPALRM output polarity can be configured through the POL bit in the RTC_CR register.

System reset, as well as low-power modes (Sleep, Stop 0, 1, 2, and Standby retention) have no influence on the wake-up timer.

33.3.11 RTC initialization and configuration

RTC Binary, BCD or Mixed mode

By default the RTC is in BCD mode (BIN = 00 in the RTC_ICSR register): the RTC_SSR register contains the subsecond field SS[15:0], clocked by ck_apre, allowing to generate a 1 Hz clock to update the calendar registers in BCD format (RTC_TR and RTC_DR).

When the RTC is configured in binary mode (BIN = 01 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar registers in BCD format (RTC_TR and RTC_DR) are not used.

When the RTC is configured in mixed mode (BIN = 10 or 11 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar is updated (1 second increment) each time the SSR[BCDU+7:0] reaches 0.

RTC register write protection

After a system reset, the RTC registers are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set to enable RTC registers write access.

After Backup domain reset, the RTC registers related to RTC initialization and RTC calibration are write-protected: Writing the unlock sequence in RTC_WPR is required to gain write access to these registers.

The following steps are required to unlock the write protection on the registers related to RTC initialization that is RTC_TR, RTC_DR, RTC_PRER, the bits INIT, BIN and BCDU in RTC_ICSR, and the bit FMT in RTC_CR:

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Those two write accesses must be performed with the privilege and security mode defined by PRIV, INITPRIV, SEC, INITSEC configuration

Writing an incorrect key reactivates the write protection.

Any write access to RTC_WPR with incorrect privilege, security settings has no effect and is ignored without any illegal access notification.

The following steps are required to unlock the write protection of the registers related to RTC calibration, that is RTC_CALR, RTC_SHIFTR, and the bits SUB1H, ADD1H, REFCKON in RTC_CR:

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Those two write accesses must be done with the privilege and security mode defined by PRIV, CALPRIV, SEC, CALSEC configuration.

Writing an incorrect key reactivates the write protection.

Any write access to RTC_WPR with the incorrect privilege, security settings has no effect and is ignored without any illegal access notification.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ICSR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ICSR register. The initialization phase mode is entered when INITF is set to 1.
   – If LPCAL=0: INITF is set around 2 RTCCLK cycles after INIT bit is set.
   – If LPCAL=1: INITF is set up to 2 ck_apre cycle after INIT bit is set.
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register, plus BIN and BCDU in the RTC_ICSR register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded.
   – If LPCAL=0: the counting restarts after 4 RTCCLK clock cycles.
   – If LPCAL=1: the counting restarts after up to 2 RTCCLK + 1 ck_apre.

When the initialization sequence is complete, the calendar starts counting. The RTC_SSR content is initialized with:

- PREDIV_S in BCD mode (BIN=00)
- 0xFFFF FFFF in binary or mixed (BCD-binary) modes (BIN=01, 10 or 11).

In BCD mode, RTC_SSR contains the value of the synchronous prescaler counter. This enables one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]). The maximum resolution allowed (30.52 µs with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption. The RTC dynamic consumption is optimized for PREDIV_A+1 being a power of 2.

Note: There is no hardware protection against incorrect calendar value initialization. Software must ensure that the initialization is performed correctly using allowed calendar BCD values.

Note: After a system reset, the application can read the INITS flag in the RTC_ICSR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its Backup domain reset default value (0x00).

Note: To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ICSR register.

Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for alarm A but can be translated in the same way for alarm B.

1. Clear ALRAE in RTC_CR to disable alarm A.
2. Program the alarm A registers (RTC_ALRMASSR/RTC_ALRMAR or RTC_ALRABINR).
3. Set ALRAE in the RTC_CR register to enable alarm A again.

Note: Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

Programming the wake-up timer

The following sequence is required to configure or change the wake-up timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wake-up timer.
2. Poll WUTWF until it is set in RTC_ICSR to make sure the access to wake-up auto-reload counter and to WUCKSEL[2:0] bits is allowed.
   - If WUCKSEL[2] = 0: WUTWF is set around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is cleared.
   - If WUCKSEL[2] = 1: WUTWF is set up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is cleared.
3. Program the wake-up auto-reload value WUT[15:0], WUTOCLR[15:0] and the wake-up clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wake-up timer restarts down-counting.
   - If WUCKSEL[2] = 0: WUTWF is cleared around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is set.
   - If WUCKSEL[2] = 1: WUTWF is cleared up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is set.

33.3.12 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB clock frequency (f_{PCLK}) must be equal to or greater than seven times the RTC clock frequency (f_{RTCCLK}). This ensures a secure behavior of the synchronization mechanism.

If the APB clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB clock frequency must never be lower than two times the RTC clock frequency.

The RSF bit is set in RTC_ICSR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every RTCCLK cycle. To ensure consistency between the 3 values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller

than 1 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode (Stop 0, 1, 2, or Standby retention), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wake-up and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to *Calendar initialization and configuration on page 1115*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 33.3.14: RTC synchronization*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (Stop 0, 1, 2, or Standby retention), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the two readings. If the values differ, repeat the process until matching results are obtained. Alternatively, the software can just compare the two results of the least-significant calendar register.

Note: While BYPSHAD = 1, instructions which read the calendar registers require one extra APB cycle to complete.

33.3.13 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ICSR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a Backup domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration register (RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers (RTC_TSSSR, RTC_TSTR and RTC_TSDR), the wake-up timer register (RTC_WUTR), the alarm A and alarm B registers (RTC_ALRMASSR/RTC_ALRMAR/RTC_ALRABINR and RTC_ALRMBSSR/RTC_ALRMBR/RTC_ALRBBINR).

In addition, when clocked by LSE, the RTC keeps on running under system reset if the reset source is different from the Backup domain reset one (refer to RCC for details about RTC

clock sources not affected by system reset). When a Backup domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

33.3.14 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the subsecond field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock with a resolution of 1 ck_apre period.

The shift operation consists in adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this delays the clock.

If at the same time the ADD1S bit is set in BCD or mixed mode, this results in adding one second and at the same time subtracting a fraction of second, so this advances the clock. ADD1S has no effect in binary mode.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

Caution: In mixed mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Caution: Before initiating a shift operation in BCD mode, the user must check that SS[15] = 0 in order to ensure that no overflow occurs. In mixed mode, the user must check that the bit SS[BCDU+8] = 0.

Caution: This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON = 1.

33.3.15 RTC reference clock detection

This feature is available only in BCD mode (BIN=00).

The update of the RTC calendar can be synchronized to a reference clock, RTC_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC_REFIN detection is enabled (REFCKON bit of RTC_CR set to 1), the calendar is still clocked by the LSE, and RTC_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck_apre periods when detecting the first reference clock edge. A smaller window of 3 ck_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the asynchronous prescaler which outputs the ck_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck_apre period detection window centered on the ck_spre edge.

When the RTC_REFIN detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:

- PREDIV_A = 0x007F
- PREVID_S = 0x00FF

Note: RTC_REFIN clock detection is not available in Standby retention mode.

33.3.16 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual ck_cal pulses).

If LPCAL=0: ck_cal = RTCCLK

If LPCAL=1: ck_cal = ck_apre

These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

RTC ultra-low-power mode

The RTC consumption can be reduced by setting the LPCAL bit in the RTC calibration register (RTC_CALR). In this case, the calibration mechanism is applied on ck_apre instead of RTCCLK. The resulting accuracy is the same, but the calibration is performed during a calibration cycle of about 2^{20} x PREDIV_A x RTCCLK pulses instead of 2^{20} RTCCLK pulses when LPCAL=0.

Smooth calibration mechanism

The smooth calibration register (RTC_CALR) specifies the number of ck_cal clock cycles to be masked during the calibration cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

Note: CALM[8:0] (RTC_CALR) specifies the number of ck_cal pulses to be masked during the calibration cycle. Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1] = 1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); CALM[2] = 1 causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[8] = 1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM permits the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to 1 effectively inserts an extra ck_cal pulse every 2^{11} ck_cal cycles, which means that 512 clocks are added during every calibration cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 ck_cal cycles can be added during the calibration cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

Caution: PREDIV_A must be greater or equal to 3.

Calibration when PREDIV_A < 3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

It is however possible to perform a calibration with PREDIV_A less than 3 in BCD mode, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 ck_cal clock cycles, which is equivalent to adding 256 clock cycles every calibration cycle. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each calibration cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S should be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (256 - CALM) / (2^{20} + CALM - 256)]$$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

Verifying the RTC calibration

It is recommended to verify the RTC calibration with LPCAL = 0, in order to have a 32-second calibration cycle.

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

  Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).
- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16- second calibration cycle period.

  In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.
- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8-second calibration cycle period.

  In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8 s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

Re-calibration on-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ICSR/INITF = 0, by using the follow process:

1. Poll the RTC_ICSR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

33.3.17 Timestamp function

Timestamp is enabled by setting the TSE bit of RTC_CR register to 1.

When TSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a timestamp event is detected on the RTC_TS pin.

When TAMPTS is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal or external tamper event is detected. Refer to *RTC control register (RTC_CR)* and refer to *Section : Timestamp on tamper event*.

When a timestamp event occurs, the timestamp flag bit (TSF) in RTC_SR register is set.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

Note: TSF is set up to 2 ck_apre cycles after the timestamp event from RTC_TS pin occurs, due to synchronization process. TSF is set up to 3 ck_apre cycles after tamper flags.

TSOVF is set up to only 1 ck_apre cycle after the event occurs. This means that if two timestamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

Caution: If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write 0 into TSF bit unless it has already read it to 1.

33.3.18 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the CALIB device output.

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the CALIB frequency is $f_{RTCCLK/64}$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV_S+1” is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the CALIB frequency is $f_{RTCCLK}/(256 * (PREDIV_A+1))$. This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = Ox7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

Note: When COSEL is cleared, the CALIB output is the output of the 6th stage of the asynchronous prescaler. If LPCAL is changed from 0 to 1, the output can be irregular (glitch...) during the LPCAL switch. If LPCAL = 1 this output is always available. If LPCAL = 0, no output is present if PREDIV_A is < 0x20.

When COSEL is set, the CALIB output is the output of the 8th stage of the synchronous prescaler.

33.3.19 Tamper and alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output TAMPALRM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_SR register.

When the TAMPOE control bit is set in the RTC_CR, all external and internal tamper flags are ORed and routed to the TAMPALRM output. If OSEL = 00 the TAMPALRM output reflects only the tampers flags. If OSEL ≠ 00, the signal on TAMPALRM provides both tamper flags and alarm A, B, or wake-up flag.

The polarity of the TAMPALRM output is determined by the POL control bit in RTC_CR so that the opposite of the selected flags bit is output when POL is set to 1.

TAMPALRM output

The TAMPALRM pin can be configured in output open drain or output push-pull using the control bit TAMPALRM_TYPE in the RTC_CR register. It is possible to apply the internal pull-up in output mode thanks to TAMPALRM_PU in the RTC_CR.

Note: Once the TAMPALRM *output is enabled, it has priority over CALIB on RTC_OUT1.*

33.4 RTC low-power modes

Table 307. Effect of low-power modes on RTC

Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the Sleep mode.
Stop 0, 1, and 2	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Stop mode.
Stop 3	The RTC state is retained, but counting operation is stopped. The RTC resumes counting when exiting Stop mode.
Standby retention	The RTC remains active when retained with RTCRSB in PWR_RC1 and the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Standby retention mode.
Standby	The RTC is powered-down and must be reinitialized after exiting the Standby mode.

The table below summarizes the RTC pins and functions capability in all modes.

Table 308. RTC pins functionality over modes

Functions	Functional in all low-power Stop(1) modes	Functional in Standby retention mode
RTC_TS	Yes	Yes
RTC_REFIN	Yes	No
RTC_OUT1	Yes	Yes
RTC_OUT2	Yes	Yes

1. Only for Stop 0, 1, and 2 modes.

33.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register or in the secure masked interrupt status register depending on its security mode configuration. The nonsecure interrupt output or the secure interrupt output is also activated.

Table 309. Nonsecure interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from low-power modes
RTC	Alarm A	ALRAF	ALRAIE and (ALRASEC=0 and SEC=0)	write 1 in CALRAF	Yes(3)
	Alarm B	ALRBF	ALRBIE and (ALRBSEC=0 and SEC=0)	write 1 in CALRBF	Yes(3)
	Timestamp	TSF	TSIE and (TSSEC=0 and SEC=0)	write 1 in CTSF	Yes(3)
	Wake-up timer	WUTF	WUTIE and (WUTSEC=0 and SEC=0)	write 1 in CWUTF	Yes(3)
	SSR underflow (reload)	SSRUF	SSRUIE and (ALRASEC=0 and SEC=0)	write 1 in CSSRUF	Yes(3)

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_MISR register.
3. When the RTC is clocked by an oscillator functional in the low-power mode.

Table 310. Secure interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from low-power modes
RTC_S	Alarm A	ALRAF	ALRAIE and (ALRASEC=1 or SEC=1)	write 1 in CALRAF	Yes(3)
	Alarm B	ALRBF	ALRBIE and (ALRBSEC=1 or SEC=1)	write 1 in CALRBF	Yes(3)
	Timestamp	TSF	TSIE and (TSSEC=1 or SEC=1)	write 1 in CTSF	Yes(3)
	Wake-up timer	WUTF	WUTIE and (WUTSEC=1 or SEC=1)	write 1 in CWUTF	Yes(3)
	SSR underflow (reload)	SSRUF	SSRUIE and (ALRASEC=1 or SEC=1)	write 1 in CSSRUF	Yes(3)

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_SMISR register.
3. When the RTC is clocked by an oscillator functional in the low-power mode.

33.6 RTC registers

Refer to *Section 1.2* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

33.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 1115* and *Reading the calendar on page 1116*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1114*.

This register can be write-protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be write-protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

33.6.2 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 1115* and *Reading the calendar on page 1116*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1114*.

This register can be write-protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be write-protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset value: 0x0000 2101 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units
000: forbidden
001: Monday
...
111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

Note: The calendar is frozen when reaching the maximum value, and can't roll over.

33.6.3 RTC subsecond register (RTC_SSR)

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

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

Bits 31:0 **SS[31:0]**: Synchronous binary counter

SS[31:16]: Synchronous binary counter MSB values

When Binary or Mixed mode is selected (BIN = 01 or 10 or 11):

SS[31:16] are the 16 MSB of the SS[31:0] free-running down-counter.

When BCD mode is selected (BIN=00):

SS[31:16] are forced by hardware to 0x0000.

SS[15:0]: Subsecond value/synchronous binary counter LSB values

When Binary mode is selected (BIN = 01 or 10 or 11):

SS[15:0] are the 16 LSB of the SS[31:0] free-running down-counter.

When BCD mode is selected (BIN=00):

SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:

Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)

SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

33.6.4 RTC initialization control and status register (RTC_ICSR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1114*.

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x0C

Backup domain reset value: 0x0000 0007

System reset: not affected except INIT, INITF, and RSF bits which are cleared to 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BCDU[2:0]			BIN[1:0]		INIT	INITF	RSF	INITS	SHPF	WUTWF	Res.	Res.
			rw	rw	rw	rw	rw	rw	r	rc_w0	r	r	r		

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to 1 when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to 0. Refer to *Re-calibration on-the-fly*.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:10 **BCDU[2:0]**: BCD update (BIN = 10 or 11)

In mixed mode when both BCD calendar and binary extended counter are used (BIN = 10 or 11), the calendar second is incremented using the SSR Least Significant Bits.

0x0: 1s calendar increment is generated each time SS[7:0] = 0
0x1: 1s calendar increment is generated each time SS[8:0] = 0
0x2: 1s calendar increment is generated each time SS[9:0] = 0
0x3: 1s calendar increment is generated each time SS[10:0] = 0
0x4: 1s calendar increment is generated each time SS[11:0] = 0
0x5: 1s calendar increment is generated each time SS[12:0] = 0
0x6: 1s calendar increment is generated each time SS[13:0] = 0
0x7: 1s calendar increment is generated each time SS[14:0] = 0

Bits 9:8 **BIN[1:0]**: Binary mode

00: Free running BCD calendar mode (Binary mode disabled).
01: Free running Binary mode (BCD mode disabled)
10: Free running BCD calendar and Binary modes
11: Free running BCD calendar and Binary modes

Bit 7 **INIT**: Initialization mode

0: Free running mode

1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER), plus BIN and BCDU fields. Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed

1: Calendar registers update is allowed

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSR, RTC_TR and RTC_DR). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF = 1), or when in bypass shadow register mode (BYPSHAD = 1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Note: RSF is not set when the frequency of the APB clock is less than twice the frequency of RTCCLK.

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (Backup domain reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

0: No shift operation is pending

1: A shift operation is pending

Bit 2 **WUTWF**: Wake-up timer write flag

This bit is set by hardware when WUT value can be changed, after the WUTE bit has been set to 0 in RTC_CR.

0: Wake-up timer configuration update not allowed

1: Wake-up timer configuration update allowed

Bits 1:0 Reserved, must be kept at reset value.

33.6.5 RTC prescaler register (RTC_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to *Calendar initialization and configuration on page 1115*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1114*.

This register can be write-protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be write-protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x10

Backup domain reset value: 0x007F 00FF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PREDIV_S[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **PREDIV_A[6:0]**: Asynchronous prescaler factor

This is the asynchronous division factor:
ck_apre frequency = RTCCLK frequency/(PREDIV_A+1)

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV_S[14:0]**: Synchronous prescaler factor

This is the synchronous division factor:
ck_spre frequency = ck_apre frequency/(PREDIV_S+1)

33.6.6 RTC wake-up timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ICSR.

This register can be protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x14

Backup domain reset value: 0x0000 FFFF

System reset: not affected

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

Bits 31:16 **WUTOCLR[15:0]**: Wake-up auto-reload output clear value

When WUTOCLR[15:0] is different from 0x0000, WUTF is set by hardware when the auto-reload down-counter reaches 0 and is cleared by hardware when the auto-reload downcounter reaches WUTOCLR[15:0].

When WUTOCLR[15:0] = 0x0000, WUTF is set by hardware when the WUT down-counter reaches 0 and is cleared by software.

Bits 15:0 **WUT[15:0]**: Wake-up auto-reload value bits

When the wake-up timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register.

When WUCKSEL[2] = 1, the wake-up timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs between WUT and (WUT + 2) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

33.6.7 RTC control register (RTC_CR)

This register is write protected. The write access procedure is described in RTC register write protection on page 1114.

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	ALRBFCLR	ALRAFCLR	TAMPOE	TAMPTS	Res.	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OUT2EN**: RTC_OUT2 output enable

With this bit set, the RTC outputs can be remapped on RTC_OUT2 as follows:
OUT2EN = 0: RTC output 2 disable
If OSEL ≠ 00 or TAMPOE = 1: TAMPALRM is output on RTC_OUT1
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT1
OUT2EN = 1: RTC output 2 enable
If (OSEL ≠ 00 or TAMPOE = 1) and COE = 0: TAMPALRM is output on RTC_OUT2
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT2
If (OSEL ≠ 00 or TAMPOE = 1) and COE = 1: CALIB is output on RTC_OUT2 and TAMPALRM is output on RTC_OUT1.

Bit 30 **TAMPALRM_TYPE**: TAMPALRM output type

0: TAMPALRM is push-pull output
1: TAMPALRM is open-drain output

Bit 29 **TAMPALRM_PU**: TAMPALRM pull-up enable

0: No pull-up is applied on TAMPALRM output
1: A pull-up is applied on TAMPALRM output

Bit 28 **ALRBFCLR**: Alarm B flag automatic clear

0: Alarm B event generates a trigger event and ALRBF must be cleared by software to allow next alarm event.
1: Alarm B event generates a trigger event. ALRBF is automatically cleared by hardware after 1 ck_apre cycle.

Bit 27 **ALRAFCLR**: Alarm A flag automatic clear

0: Alarm A event generates a trigger event and ALRAF must be cleared by software to allow next alarm event.
1: Alarm A event generates a trigger event. ALRAF is automatically cleared by hardware after 1 ck_apre cycle.

Bit 26 **TAMPOE**: Tamper detection output enable on TAMPALRM
0: The tamper flag is not routed on TAMPALRM
1: The tamper flag is routed on TAMPALRM, combined with the signal provided by OSEL and with the polarity provided by POL.

Bit 25 **TAMPTS**: Activate timestamp on tamper detection event
0: Tamper detection event does not cause a RTC timestamp to be saved
1: Save RTC timestamp on tamper detection event
TAMPTS is valid even if TSE = 0 in the RTC_CR register. Timestamp flag is set up to 3 ck_apre cycles after the tamper flags.
Note: TAMPTS must be cleared before entering RTC initialization mode.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **COE**: Calibration output enable
This bit enables the CALIB output
0: Calibration output disabled
1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection
These bits are used to select the flag to be routed to TAMPALRM output.
00: Output disabled
01: Alarm A output enabled
10: Alarm B output enabled
11: Wake-up output enabled

Bit 20 **POL**: Output polarity
This bit is used to configure the polarity of TAMPALRM output.
0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).
1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).

Bit 19 **COSEL**: Calibration output selection
When COE = 1, this bit selects which signal is output on CALIB.
0: Calibration output is 512 Hz
1: Calibration output is 1 Hz
These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A = 127 and PREDIV_S = 255). Refer to *Section 33.3.18: Calibration clock output*.

Bit 18 **BKP**: Backup
This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: *S*ubtract 1 hour (winter time change)
When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.
Setting this bit has no effect when current hour is 0.
0: No effect
1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)
When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.
0: No effect
1: Adds 1 hour to the current time. This can be used for summer time change
Note: ADD1H is ignored when SUB1H is set.

Bit 15 **TSIE**: Timestamp interrupt enable
0: Timestamp interrupt disable
1: Timestamp interrupt enable

Bit 14 **WUTIE**: Wake-up timer interrupt enable
0: Wake-up timer interrupt disabled
1: Wake-up timer interrupt enabled

Bit 13 **ALRBIE**: Alarm B interrupt enable
0: Alarm B interrupt disable
1: Alarm B interrupt enable

Bit 12 **ALRAIE**: Alarm A interrupt enable
0: Alarm A interrupt disabled
1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp enable
0: timestamp disable
1: timestamp enable

Bit 10 **WUTE**: Wake-up timer enable
0: Wake-up timer disabled
1: Wake-up timer enabled
Note: When the wake-up timer is disabled, wait for WUTWF = 1 before enabling it again.

Bit 9 **ALRBE**: Alarm B enable
0: Alarm B disabled
1: Alarm B enabled

Bit 8 **ALRAE**: Alarm A enable
0: Alarm A disabled
1: Alarm A enabled

Bit 7 **SSRUIE**: SSR underflow interrupt enable
0: SSR underflow interrupt disabled
1: SSR underflow interrupt enabled

Bit 6 **FMT**: Hour format
0: 24 hour/day format
1: AM/PM hour format

Bit 5 **BYPSHAD**: Bypass the shadow registers
0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.
1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.
Note: If the frequency of the APB clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to 1.

Bit 4 **REFCKON**: RTC_REFIN reference clock detection enable (50 or 60 Hz)
0: RTC_REFIN detection disabled
1: RTC_REFIN detection enabled
Note: BIN must be 0x00 and PREDIV_S must be 0x00FF.

Bit 3 **TSEDGE**: Timestamp event active edge
0: RTC_TS input rising edge generates a timestamp event
1: RTC_TS input falling edge generates a timestamp event
TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: ck_wut wake-up clock selection
000: RTC/16 clock is selected
001: RTC/8 clock is selected
010: RTC/4 clock is selected
011: RTC/2 clock is selected
10x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU.
11x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU. Furthermore, 2^{16} is added to the WUT counter value.

Note: Bits 6 and 4 of this register can be written in initialization mode only (RTC_ICSR/INITF = 1).

WUT = wake-up unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ICSR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it may mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

33.6.8 RTC privilege mode control register (RTC_PRIVCFGR)

This register can be written only when the APB access is privileged. This register can be write-protected, or each bit of this register can be individually write-protected against nonsecure access depending on the RTC_SECCFGR configuration (refer to *Section 33.3.5: RTC privilege protection modes*).

Address offset: 0x1C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV	INIT PRIV	CAL PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS PRIV	WUT PRIV	ALRB PRIV	ALRA PRIV
rw	rw	rw										rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PRIV**: RTC privilege protection

0: All RTC registers can be written when the APB access is privileged or unprivileged, except the registers protected by other privilege protection bits.

1: All RTC registers can be written only when the APB access is privileged.

Bit 14 **INITPRIV**: Initialization privilege protection

0: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is privileged or unprivileged.

1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is privileged.

Bit 13 **CALPRIV**: Shift register, daylight saving, calibration and reference clock privilege protection

0: Shift register, daylight saving, calibration and reference clock can be written when the APB access is privileged or unprivileged.

1: Shift register, daylight saving, calibration and reference clock can be written only when the APB access is privileged.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **TSPRIV**: Timestamp privilege protection

0: RTC Timestamp configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC Timestamp configuration and interrupt clear can be written only when the APB access is privileged.

Bit 2 **WUTPRIV**: Wake-up timer privilege protection

0: RTC wake-up timer configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC wake-up timer configuration and interrupt clear can be written only when the APB access is privileged.

Bit 1 **ALRBPRIV**: Alarm B privilege protection

0: RTC Alarm B configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC Alarm B configuration and interrupt clear can be written only when the APB access is privileged.

Bit 0 **ALRAPRIV**: Alarm A and SSR underflow privilege protection

0: RTC Alarm A and SSR underflow configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC Alarm A and SSR underflow configuration and interrupt clear can be written only when the APB access is privileged.

Note: Refer to Section 33.3.5: RTC privilege protection modes for details on the read protection.

33.6.9 RTC secure configuration register (RTC_SECCFGR)

This register can be written only when the APB access is secure.

This register can be globally write-protected, or each bit of this register can be individually write-protected against unprivileged access depending on the RTC_PRIVCFGR configuration (refer to *Section 33.3.5: RTC privilege protection modes*).

Address offset: 0x20

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC	INIT SEC	CAL SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS SEC	WUT SEC	ALRB SEC	ALRA SEC
rw	rw	rw										rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **SEC**: RTC global protection

0: All RTC registers can be written when the APB access is secure or nonsecure, except the registers protected by other secure protection bits.

1: All RTC registers can be written only when the APB access is secure.

Bit 14 **INITSEC**: Initialization protection

0: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is secure or nonsecure.

1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is secure.

Bit 13 **CALSEC**: Shift register, daylight saving, calibration and reference clock protection

0: Shift register, daylight saving, calibration and reference clock can be written when the APB access is secure or nonsecure.

1: Shift register, daylight saving, calibration and reference clock can be written only when the APB access is secure.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **TSSEC**: Timestamp protection

0: RTC timestamp configuration and interrupt clear can be written when the APB access is secure or nonsecure.

1: RTC timestamp configuration and interrupt clear can be written only when the APB access is secure.

Bit 2 **WUTSEC**: Wake-up timer protection

0: RTC wake-up timer configuration and interrupt clear can be written when the APB access is secure or nonsecure.

1: RTC wake-up timer configuration and interrupt clear can be written only when the APB access is secure.

Bit 1 **ALRBSEC**: Alarm B protection

0: RTC alarm B configuration and interrupt clear can be written when the APB access is secure or nonsecure.

1: RTC alarm B configuration and interrupt clear can be written only when the APB access is secure.

Bit 0 **ALRASEC**: Alarm A and SSR underflow protection

0: RTC alarm A and SSR underflow configuration and interrupt clear can be written when the APB access is secure or nonsecure.

1: RTC alarm A and SSR underflow configuration and interrupt clear can be written only when the APB access is secure.

Note: Refer to Section 33.3.4: RTC secure protection modes for details on the read protection.

33.6.10 RTC write protection register (RTC_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **KEY[7:0]**: Write protection key

This byte is written by software.

Reading this byte always returns 0x00.

Refer to *RTC register write protection* for a description of how to unlock RTC register write protection.

33.6.11 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1114*.

This register can be write-protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be write-protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x28

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW 16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CALP**: Increase frequency of RTC by 488.5 ppm
0: No RTCCLK pulses are added.
1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency increased by 488.5 ppm).
This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. if the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows:
(512 × CALP) - CALM.
Refer to *Section 33.3.16: RTC smooth digital calibration*.

Bit 14 **CALW8**: Use an 8-second calibration cycle period
When CALW8 is set to 1, the 8-second calibration cycle period is selected.
Note: CALM[1:0] are stuck at 00 when CALW8 = 1. Refer to Section 33.3.16: RTC smooth digital calibration.

Bit 13 **CALW16**: Use a 16-second calibration cycle period
When CALW16 is set to 1, the 16-second calibration cycle period is selected. This bit must not be set to 1 if CALW8 = 1.
Note: CALM[0] is stuck at 0 when CALW16 = 1. Refer to Section 33.3.16: RTC smooth digital calibration.

Bit 12 **LPCAL**: RTC low-power mode

0: Calibration window is 2^{20} RTCCLK, which is a high-consumption mode. This mode must be set only when less than 32s calibration window is required.

1: Calibration window is 2^{20} ck_apre, which is the required configuration for ultra-low consumption mode.

Bits 11:9 Reserved, must be kept at reset value.

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of 2^{20} RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See *Section 33.3.16: RTC smooth digital calibration on page 1119*.

33.6.12 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1114*.

This register can be protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBFS[14:0]														
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 Reserved, must be kept at reset value.

Bits 14:0 **SUBFS[14:0]**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (PREDIV_S + 1)

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

Advance (seconds) = (1 - (SUBFS / (PREDIV_S + 1))).

In mixed BCD-binary mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF = 1 to be sure that the shadow registers have been updated with the shifted time.

33.6.13 RTC timestamp time register (RTC_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	r	r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation

0: AM or 24-hour format

1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

33.6.14 RTC timestamp date register (RTC_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
r	r	r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **WDU[2:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

33.6.15 RTC timestamp subsecond register (RTC_TSSSR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when the TSF bit is reset.

This register can be protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x38

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 SS[31:0]: Subsecond value/synchronous binary counter values

SS[31:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

33.6.16 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAE is reset in RTC_CR register.

This register can be protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x40

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm A date mask
0: Alarm A set if the date/day match
1: Date/day don’t care in alarm A comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don’t care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm A hours mask
0: Alarm A set if the hours match
1: Hours don’t care in alarm A comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm A minutes mask
0: Alarm A set if the minutes match
1: Minutes don’t care in alarm A comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm A seconds mask
0: Alarm A set if the seconds match
1: Seconds don’t care in alarm A comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

33.6.17 RTC alarm A subsecond register (RTC_ALRMASSR)

This register can be written only when ALRAE is reset in RTC_CR register.

This register can be protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x44

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSCLR	Res.	MASKSS[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw		rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)

0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.

1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRABINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRABINR.SS[31:0].

Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit

0: No comparison on subseconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[31:1] are don't care in Alarm A comparison. Only SS[0] is compared.

2: SS[31:2] are don't care in Alarm A comparison. Only SS[1:0] are compared.

...

31: SS[31] is don't care in Alarm A comparison. Only SS[30:0] are compared.

From 32 to 63: All 32 SS bits are compared and must match to activate alarm.

Note: In BCD mode (BIN=00) the overflow bits of the synchronous counter (bits 31:15) are never compared. These bits can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Subseconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

This field is the mirror of SS[14:0] in the RTC_ALRABINR, and so can also be read or written through RTC_ALRABINR.

Note: SS[3:0] must be 0000 when SSCLR is set with ATCKSEL[3] = 1 in TAMP_ATCR1.

33.6.18 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBE is reset in RTC_CR register.

This register can be protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x48

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WD SEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm B date mask
0: Alarm B set if the date and day match
1: Date and day don’t care in alarm B comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don’t care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm B hours mask
0: Alarm B set if the hours match
1: Hours don’t care in alarm B comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm B minutes mask
0: Alarm B set if the minutes match
1: Minutes don’t care in alarm B comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm B seconds mask
0: Alarm B set if the seconds match
1: Seconds don't care in alarm B comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

33.6.19 RTC alarm B subsecond register (RTC_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC_CR register.

This register can be protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x4C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSCLR	Res.	MASKSS[5:4]		MASKSS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw		rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)
0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.
1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRBBINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRBBINR.SS[31:0].
Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit
0: No comparison on subseconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
1: SS[31:1] are don't care in Alarm B comparison. Only SS[0] is compared.
2: SS[31:2] are don't care in Alarm B comparison. Only SS[1:0] are compared.
...
31: SS[31] is don't care in Alarm B comparison. Only SS[30:0] are compared.
From 32 to 63: All 32 SS bits are compared and must match to activate alarm.
Note: In BCD mode (BIN=00)The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Subseconds value
This value is compared with the contents of the synchronous prescaler counter to determine if alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.
This field is the mirror of SS[14:0] in the RTC_ALRBBINR, and so can also be read or written through RTC_ALRBBINR.
Note: SS[3:0] must be 0000 when SSCLR is set with ATCKSEL[3] = 1 in TAMP_ATCR1.

33.6.20 RTC status register (RTC_SR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x50

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUF	Res.	TSOVF	TSF	WUTF	ALRBF	ALRAF
									r		r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUF**: SSR underflow flag
This flag is set by hardware when the SSR is reloaded with 0xFFFF FFFF after reaching 0. SSRUF is not set when SSCLR = 1.
Note: SSRUF is not an error event as SSR counter is a free-running down-counter with automatic reload. SSRUF is set only in binary and mixed modes. It is not set in BCD mode.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TSOVF**: Timestamp overflow flag

This flag is set by hardware when a timestamp event occurs while TSF is already set.

It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSF**: Timestamp flag

This flag is set by hardware when a timestamp event occurs.

Note: TSF is not set if TAMPTS = 1 and the tamper flag is read during the 3 ck_apre cycles following tamper event. Refer to Timestamp on tamper event for more details.

Bit 2 **WUTF**: Wake-up timer flag

This flag is set by hardware when the wake-up auto-reload counter reaches 0.

If WUTOCLR[15:0] is different from 0x0000, WUTF is cleared by hardware when the wake-up auto-reload counter reaches WUTOCLR value.

If WUTOCLR[15:0] is 0x0000, WUTF must be cleared by software.

This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBF**: Alarm B flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm B register (RTC_ALRMBR).

Bit 0 **ALRAF**: Alarm A flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm A register (RTC_ALRMAR).

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

33.6.21 RTC nonsecure masked interrupt status register (RTC_MISR)

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x54

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	Res.	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
									r		r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUMF**: SSR underflow nonsecure masked flag

This flag is set by hardware when the SSR underflow nonsecure interrupt occurs.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TSOVMF**: Timestamp overflow nonsecure masked flag
This flag is set by hardware when a timestamp interrupt occurs while TSMF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp nonsecure masked flag
This flag is set by hardware when a timestamp nonsecure interrupt occurs.

Bit 2 **WUTMF**: Wake-up timer nonsecure masked flag
This flag is set by hardware when the wake-up timer nonsecure interrupt occurs.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B nonsecure masked flag
This flag is set by hardware when the alarm B nonsecure interrupt occurs.

Bit 0 **ALRAMF**: Alarm A masked flag
This flag is set by hardware when the alarm A nonsecure interrupt occurs.

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

33.6.22 RTC secure masked interrupt status register (RTC_SMISR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x58

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	Res.	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
									r		r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUMF**: SSR underflow secure masked flag
This flag is set by hardware when the SSR underflow secure interrupt occurs.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TSOVMF**: Timestamp overflow interrupt secure masked flag
This flag is set by hardware when a timestamp secure interrupt occurs while TSMF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp interrupt secure masked flag
This flag is set by hardware when a timestamp secure interrupt occurs.

Bit 2 **WUTMF**: Wake-up timer interrupt secure masked flag
This flag is set by hardware when the wake-up timer secure interrupt occurs.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B interrupt secure masked flag
This flag is set by hardware when the alarm B secure interrupt occurs.

Bit 0 **ALRAMF**: Alarm A interrupt secure masked flag
This flag is set by hardware when the alarm A secure interrupt occurs.

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

33.6.23 RTC status clear register (RTC_SCR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x5C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	Res.	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
									w		w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CSSRUF**: Clear SSR underflow flag
Writing ‘1’ in this bit clears the SSRUF in the RTC_SR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CTSOVF**: Clear timestamp overflow flag

Writing 1 in this bit clears the TSOVF bit in the RTC_SR register.

It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **CTSF**: Clear timestamp flag

Writing 1 in this bit clears the TSF bit in the RTC_SR register.

Bit 2 **CWUTF**: Clear wake-up timer flag

Writing 1 in this bit clears the WUTF bit in the RTC_SR register.

Bit 1 **CALRBF**: Clear alarm B flag

Writing 1 in this bit clears the ALRBF bit in the RTC_SR register.

Bit 0 **CALRAF**: Clear alarm A flag

Writing 1 in this bit clears the ALRAF bit in the RTC_SR register.

33.6.24 RTC alarm A binary mode register (RTC_ALRABINR)

This register can be written only when ALRAE is reset in RTC_CR register.

This register can be protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x70

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value

This value is compared with the contents of the synchronous counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMASSR, and so can also be read or written through RTC_ALRMASSR.

Note: SS[3:0] must be 0000 when SSCLR is set with ATCKSEL[3] = 1 in TAMP_ATCR1.

33.6.25 RTC alarm B binary mode register (RTC_ALRBBINR)

This register can be written only when ALRBE is reset in RTC_CR register.

This register can be protected against nonsecure access. Refer to *Section 33.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 33.3.5: RTC privilege protection modes*.

Address offset: 0x74

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value

This value is compared with the contents of the synchronous counter to determine if Alarm Bis to be activated. Only bits 0 up MASKSS-1 are compared.

SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMBSSR, and so can also be read or written through RTC_ALRMBSSR.

Note: SS[3:0] must be 0000 when SSCLR is set with ATCKSEL[3] = 1 in TAMP_ATCR1.

33.6.26 RTC register map

Table 311. RTC register map and reset values

Offset	Register name	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
0x00	RTC_TR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT [1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04	RTC_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]				WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1			0	0	0	0	0	1
0x08	RTC_SSR	SS[31:16]																SS[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	RTC_ICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF	Res.	Res.	Res.	BCDU [2:0]			BIN [1:0]		INIT	INITF	RSF	INITS	SHPF	WUT WF	Res.	Res.
	Reset value																0				0	0	0	0	0	0	0	0	0	0	1		
0x10	RTC_PRER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]							PREDIV_S[14:0]															
	Reset value										1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	RTC_WUTR	WUTOCLR[15:0]																WUT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x18	RTC_CR	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	ALRBFCLR	ALRAFCLR	TAMPOE	TAMPTS	Res.	COE	O SEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCK SEL[2:0]		
	Reset value	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	RTC_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	INITPRIV	CALPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSPRIV	WUTPRIV	ALRBPRIV	ALRAPRIV
	Reset value																	0	0	0										0	0	0	0
0x20	RTC_ SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC	INITSEC	CALSEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSSEC	WUTSEC	ALRBSEC	ALRASEC
	Reset value																	0	0	0										0	0	0	0
0x24	RTC_WPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	RTC_ CALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
	Reset value																	0	0	0	0				0	0	0	0	0	0	0	0	0
0x2C	RTC_SHIFTR	ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 311. RTC register map and reset values (continued)

Offset	Register name	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
0x30	RTC_TSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x34	RTC_TSDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		WDU[2:0]		MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value																		0	0	0	0	0	0	0			0	0	0	0	0	0
0x38	RTC_TSSSR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	RTC_ALRMAR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	RTC_ ALRMASSR	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	RTC_ALRMBR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	RTC_ ALRMBSSR	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	RTC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUF	Res.	TSOVF	TSF	WUTF	ALRBF	ALRAF
	Reset value																										0		0	0	0	0	0
0x54	RTC_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	Res.	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																										0		0	0	0	0	0
0x58	RTC_SMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	Res.	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																										0		0	0	0	0	0
0x5C	RTC_SCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	Res.	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
	Reset value																										0		0	0	0	0	0
0x70	RTC_ ALRABINR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74	RTC_ ALRBBINR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

34 Tamper and backup registers (TAMP)

34.1 TAMP introduction

The anti-tamper detection circuit is used to protect sensitive data from external attacks. 32 32-bit backup registers are retained in all low-power modes except Standby. The backup registers, as well as other secrets in the device, are protected by this anti-tamper detection circuit with 4 tamper pins and 9 internal tampers. The external tamper pins can be configured for edge detection, or level detection with or without filtering, or active tamper which increases the security level by auto checking that the tamper pins are not externally opened or shorted.

34.2 TAMP main features

- A tamper detection can optionally erase the backup registers, SRAM2, cache, and cryptographic peripherals. The device resources protected by tamper are named “device secrets”.
- 32×32-bit backup registers
- Up to 4 tamper pins for 4 external tamper detection events:
  - Active tamper mode: continuous comparison between tamper output and input to protect from physical open-short attacks.
  - Flexible active tamper I/O management: from 2 meshes (each input associated to its own exclusive output) to 3 meshes (single output shared for up to 3 tamper inputs)
  - Passive tampers: ultra-low power edge or level detection with internal pull-up hardware management.
  - Configurable digital filter.
- 9 internal tamper events to protect against transient or environmental perturbation attacks
- Each tamper can be configured in two modes:
  - Confirmed mode: immediate erase of secrets on tamper detection, including backup registers erase
  - Potential mode: most of the secrets erase following a tamper detection are launched by software
- Any tamper detection can generate a RTC timestamp event.
- TrustZone support:
  - Tamper secure or nonsecure configuration.
  - Backup registers configuration in 3 configurable-size areas:

    1 read/write secure area.

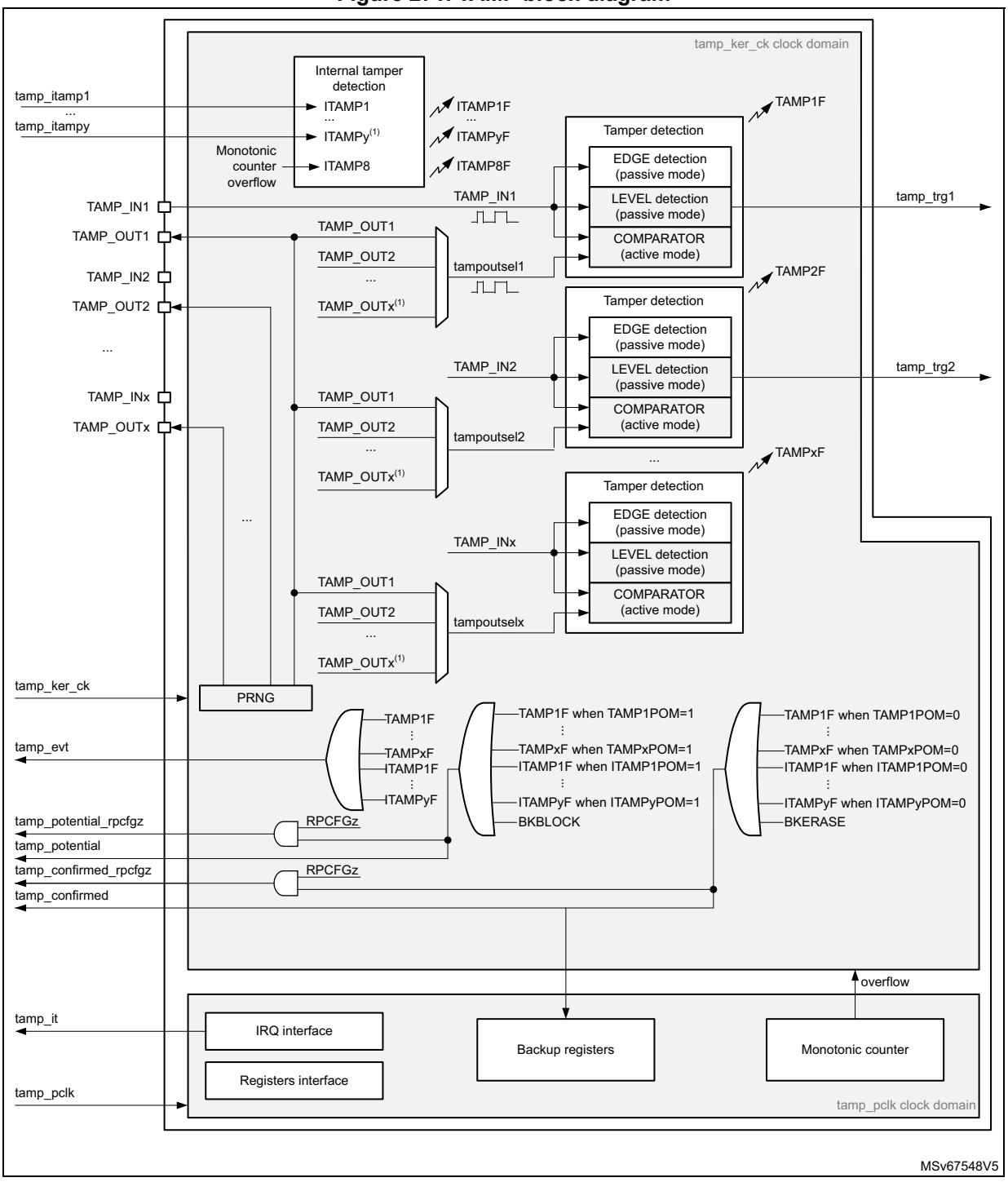
    1 write secure/read nonsecure area.

    1 read/write nonsecure area.
  - Boot hardware key for secure AES, stored in backup registers, protected against read and write access.
- Tamper configuration and backup registers privilege protection
- Monotonic counter.

34.3 TAMP functional description

34.3.1 TAMP block diagram

Figure 271. TAMP block diagram



1. The number of external and internal tampers depends on products.

34.3.2 TAMP pins and internal signals

Table 312. TAMP input/output pins

Pin name	Signal type	Description
TAMP_INx (x = pin index)	Input	Tamper input pin
TAMP_OUTx (x = pin index)	Output	Tamper output pin (active mode only)

Table 313. TAMP internal input/output signals

Internal signal name	Signal type	Description
tamp_ker_ck	Input	TAMP kernel clock, connected to rtc_ker_ck and also named RTCCLK in this document
tamp_pclk	Input	TAMP APB clock, connected to rtc_pclk
tamp_itamp[y] (y = signal index)	Inputs	Internal tamper event sources
tamp_tzen	Input	TAMP TrustZone enabled
tamp_evt	Output	Tamper event detection flag (internal or external tamper), whatever confirmed or potential mode configuration.
tamp_potential	Output	Potential tamper detection signal, used for device secrets(1) protection. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in potential mode. – or a software request is done by writing BKBLOCK to 1
tamp_confirmed	Output	Confirmed tamper detection signal, used for device secrets(1) protection. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in confirmed mode. – or a software request is done by writing BKERASE to 1
tamp_potential_rpcfgz (z = signal index)	Output	Potential tamper detection signal generated only when RPCFGz = 1. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in potential mode. – or a software request is done by writing BKBLOCK to 1

Table 313. TAMP internal input/output signals (continued)

Internal signal name	Signal type	Description
tamp_confirmed_rpcfgz (z = signal index)	Output	Confirmed tamper detection signal generated only when RPCFGz = 1. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in confirmed mode. – or a software request is done by writing BKERASE to 1
tamp_it	Output	TAMP interrupt (refer to *Section 34.5: TAMP interrupts* for details)
tamp_trg[x] (x = signal index)	Output	Tamper detection trigger

1. Refer to *Table 314: TAMP interconnection*.

The TAMP kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). The TAMP kernel clock is required only for external tamper inputs level with filtering, and for external active tamper detection modes. Internal tampers detection and external tampers inputs edge detection are functional without requiring any kernel clock.

Read and write access to backup registers and all other TAMP registers are also functional without any kernel clock (only APB clock is needed).

Some detections modes are not available in some low-power modes depending on the selected clock (refer to for more details.

Table 314. TAMP interconnection

Signal name	Source/Destination
tamp_tzen	From FLASH option bytes: TZEN
tamp_evt	rtc_tamp_evt used to generate a timestamp event
tamp_potential	The tamp_potential signal is used to block the read and write accesses to the device secrets listed hereafter: – backup registers
tamp_confirmed	The tamp_confirmed signal is used to erase the device secrets listed hereafter: – backup registers The device secrets access is blocked when erase is ongoing.
tamp_potential_rpcfg1	When the bit RPCFG1 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg1 signal is used to block the read and write accesses to the device secrets listed hereafter: – SRAM2

Table 314. TAMP interconnection (continued)

Signal name	Source/Destination
tamp_confirmed_rpcfg1	When the bit RPCFG1 is set in the TAMP_RPCFGR, the tamp_erase_rpcfg1 signal is used to erase the device secrets listed hereafter: – SRAM2 The device secrets access is blocked when erase is on-going.
tamp_potential_rpcfgz (z = 3 to 5)	When the bit RPCFGz is set in the TAMP_RPCFGR, the tamp_potential_rpcfgz signal is used to erase the device secrets listed hereafter: – z = 3: ICACHE – z = 4: AES, HASH and OTFDEC – z = 5: PKA SRAM The device secrets access is blocked when erase is on-going.
tamp_confirmed_rpcfgz (z = 3 to 5)	When the bit RPCFGz is set in the TAMP_RPCFGR, the tamp_erase_rpcfgz signal is used to erase the device secrets listed hereafter: – z = 3: ICACHE – z = 4: AES, HASH and OTFDEC – z = 5: PKA SRAM The device secrets access is blocked when erase is on-going.
tamp_itamp3	LSE monitoring (LSECSS)
tamp_itamp5	RTC calendar overflow (rtc_calovf)
tamp_itamp6	JTAG/SWD access when RDP > 0
tamp_itamp7	ADC4 (adc4_awd1) analog watchdog monitoring 1
tamp_itamp8(1)	Monotonic counter 1 overflow
tamp_itamp9	Cryptographic peripherals fault (AES or PKA or TRNG)
tamp_itamp11	IWDG reset when tamper flag is set (potential tamper timeout)
tamp_itamp12	ADC4 (adc4_awd2) analog watchdog monitoring 2
tamp_itamp13	ADC4 (adc4_awd3) analog watchdog monitoring 3

1. This signal is generated in the TAMP peripheral.

The TZEN option bit is used to activate TrustZone in the device.

TZEN = 1: TrustZone activated.

TZEN = 0: TrustZone disabled.

When TrustZone is disabled, the APB access to the TAMP registers are nonsecure.

34.3.3 GPIOs controlled by the RTC and TAMP

Refer to *Section 33.3.3: GPIOs controlled by the RTC and TAMP*.

34.3.4 TAMP register write protection

After system reset, the TAMP registers (including backup registers) are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable TAMP registers write access.

34.3.5 TAMP secure protection modes

By default after a backup domain power-on reset, all TAMP registers can be read or written in both secure and nonsecure modes, except for the TAMP secure configuration register (TAMP_SECCFGR) which can be written in secure mode only. The TAMP protection configuration is not affected by a system reset.

- When the TAMPSEC bit is set in the TAMP_SECCFGR register:
  - Writing the TAMP registers is possible only in secure mode, except for the backup registers and monotonic counters which have their own protection setting.
  - Reading TAMP_SECCFGR, TAMP_PRIVCFGR and TAMP_MISR is always possible in secure and nonsecure modes. All the other TAMP registers can be read only in secure mode, except for the backup registers and monotonic counters which have their own protection setting.
- When the CNT1SEC bit is set in the TAMP_SECCFGR register: the TAMP_COUNT1R can be read and written only in secure mode, and CNT1PRIV can be written only in secure mode.
- When the BKPRWSEC is set in the TAMP_SECCFGR register: the BKPRWPRIV can be written only in secure mode.
- When the BKPWSEC is set in the TAMP_SECCFGR register: the BKPRWPRIV and BKPWPRIV can be written only in secure mode.

A nonsecure access to a secure-protected register is denied:

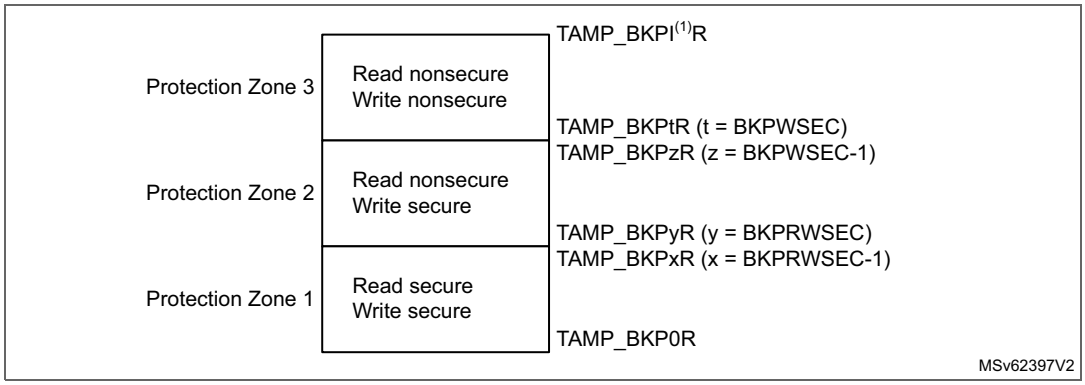
- There is no bus error generated.
- A notification is generated through a flag/interrupt in the TZIC (TrustZone illegal access controller).
- When write protected, the bits are not written.
- When read protected they are read as 0.

As soon as at least one function is configured to be secured, the TAMP reset and clock control is also secured in the RCC.

34.3.6 Backup registers protection zones

The backup registers protection is configured thanks to BKPRWSEC[7:0] and BKPWSEC[7:0] (refer to the figure below):

Figure 272. Backup registers protection zones



1. I = last backup register index

34.3.7 TAMP privilege protection modes

By default after a backup domain power-on reset, all TAMP registers can be read or written in both privileged and unprivileged modes, except for the TAMP privilege configuration register (TAMP_PRIVCFGR) which can be written in privilege mode only. The TAMP protection configuration is not affected by a system reset.

- When the TAMPPRIV bit is set in the TAMP_PRIVCFGR register:
  - Writing the TAMP registers is possible only in privilege mode, except for the backup registers and the monotonic counter which have their own protection setting.
  - Reading TAMP_SECCFGR, TAMP_PRIVCFGR is always possible in privilege and unprivilege modes. All the other TAMP registers can be read only in privileged mode, except for the backup registers and the monotonic counter which have their own protection setting.
- When the CNT1PRIV bit is set in the TAMP_PRIVCFGR register: the TAMP_COUNT1R can be read and written only in privilege mode, and CNT1SEC can be written only in privilege mode.
- When the BKPRWPRIV is set in the TAMP_PRIVCFGR register: the BKPWSEC can be written only in privilege mode.
- When the BKPWPRIV is set in the TAMP_PRIVCFGR register: the BKPRWSEC and BKPWSEC can be written only in privilege mode.

The backup registers protection is configured thanks to BKPRWSEC[7:0] and BKPRWPRIV for the protection zone 1, and thanks to BKPRWSEC[7:0], BKPWSEC[7:0] and BKPWPRIV for the protection zone 2 (refer to *Figure 272*).

A unprivileged access to a privileged-protected register is denied:

- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

For this purpose, these registers must belong to the Protection Zone 1: BKPRWSEC must be greater or equal to 8.

34.3.8 Tamper detection

The tamper detection main purpose is to protect the device secrets from device external attacks. The detection is made on events on TAMP_INx (x = pin index) I/Os, or on internal monitors detecting out-of-range device conditions.

The tamper detection can be configured for the following purposes:

- erase the backup registers and other device secrets stored in SRAMs or peripherals listed in *Table 314: TAMP interconnection*. The device secrets list is configurable thanks to *TAMP resources protection configuration register (TAMP_RPCFGR)*.
- block the read/write access to the backup registers and other device secrets stored in SRAMs or peripherals listed in *Table 314: TAMP interconnection*. The device secrets list is configurable thanks to TAMP_RPCFGR.
- generate an interrupt, capable to wake-up from low-power modes
- generate a hardware trigger for the low-power timers, or a RTC timestamp event

The external I/Os tamper detection supports 2 main configurations:

- Passive mode: TAMP_INx I/Os are monitored and a tamper is detected either on edge or on level.
- Active mode: TAMP_INx (x = pin index) is continuously compared with TAMP_OUTy (y = pin index) allowing open-short detection.

A digital filter can be applied on external tamper detection to avoid false detection. In addition, it is possible to configure each tamper source in potential mode, so that the secrets erase is not launched by hardware on tamper detection. The secrets erase can then be launched by software after software checks.

34.3.9 TAMP backup registers and other device secrets erase

The backup registers (TAMP_BKPxR) are not reset by system reset or when the device wakes up from Standby retention mode.

The backup registers and the other device secrets are not reset when the corresponding mask is set (TAMPxMSK=1 in the TAMP_CR2 register).

Note: The backup registers are also erased when the readout protection of the flash is changed from level 1 to level 0.

Tamper detection – confirmed mode

The confirmed mode is selected for TAMPx (external tamper x) when TAMPxPOM = 0 in the TAMP_CR2 register. The confirmed mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 0 in the TAMP_CR3 register. The effects of a tamper detection in confirmed mode are described with tamp_confirmed and tamp_confirmed_rpcfgx signals in the *Table 314: TAMP interconnection*.

This mode is selected to erase automatically the device secrets when the tamper is detected.

Tamper detection – potential mode

The potential mode is selected for TAMPx (external tamper x) when TAMPxPOM = 1 in the TAMP_CR2 register. The potential tamper mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 1 in the TAMP_CR3 register. The effects of a tamper detection in potential mode are described with tamp_potential and tamp_potential_rpcfgx signals in the *Table 314: TAMP interconnection*.

This mode is selected to avoid irreversible erasure of some device secrets when the tamper is detected. In this mode, some device secrets are not erased when the corresponding tamper event is detected. In addition, the read and write accesses to these device secrets are blocked as soon as the tamper detection flag is set in potential mode, until this flag is cleared by setting the corresponding clear flag in the TAMP_SCR register. Therefore the software can perform some checks to discriminate false from true tampers, and decide to launch secrets erase only in case of the potential tamper is confirmed to be a true tamper. The device secrets are erased by software by setting the BKERASE bit in the TAMP_CR2 register.

Potential tamper to confirmed tamper timeout

Some internal tampers generate a tamper event if the independent watchdog reset occurs when another tamper flag is set (refer to *Table 314: TAMP interconnection*). The IWDG tamper must be configured with ITAMPxPOM = 0. This permits the erasure of device secrets to be forced by hardware after a timeout, in case the previous tamper event was in potential mode. This is equivalent to change the “potential tamper” into “confirmed tamper” if a watchdog reset occurs before any software decision following the potential tamper event.

Device resources protection configuration

Some device resources can be configured in order to be included to the list of the device secrets protected by tamper detection.

When RPCFGz = 0 in the TAMP_RPCFGR, the device resource associated to RPCFGz is not protected by the TAMP peripheral:

- It is not affected by tamper detection (whatever confirmed or potential mode)
- It is not affected by BKERASE software command
- It is not affected by BKBLOCK software command

When RPCFGz = 1 in the TAMP_RPCFGR, the device resource associated to RPCFGz is protected by the TAMP peripheral:

- It is affected by confirmed tamper detection and BKERASE software command, as described with tamp_confirmed_rpcfgz signal in *Table 314: TAMP interconnection*
- It is affected by potential tamper detection and BKBLOCK software command, as described with tamp_potential_rpcfgz signal in *Table 314: TAMP interconnection*

Table 315. Device resource x tamper protection

-	Potential tamper or BKBLOCK	Confirmed tamper or BKERASE
RPCFGx = 0	No effect on device resource x	No effect on device resource x
RPCFGx = 1	Device secret x protected as described by tamp_potential_rpcfgx(1)	Device secret x protected as described by tamp_confirmed_rpcfgx(1)

1. Refer to *Table 314: TAMP interconnection*.

Device secrets access blocked by software

By default, the device secrets can be accessed by the application, except if a tamper event flag is detected: the device secrets access is not possible as long as a tamper flag is set.

It is possible to block the access to the device secrets by software, by setting the BKBLOCK bit of the TAMP_CR2 register. The device secrets access is possible only when BKBLOCK = 0 and no tamper flag is set.

34.3.10 Tamper detection configuration and initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the TAMP_CR register.

Each TAMP_INx tamper detection input is associated with a flag TAMPxF in the TAMP_SR register.

By setting the TAMPxIE bit in the TAMP_IER register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPxIE is not allowed when the corresponding TAMPxMSK is set.

Trigger output generation on tamper event

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMSK bit is cleared in TAMP_CR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMSK bit is set, the TAMPxF flag is masked, and kept cleared in TAMP_SR register. This configuration permits the low-power timers to be triggered automatically in Stop 0, 1, and 2 mode, without requiring the system wake-up to perform the TAMPxF clearing. In this case, the backup registers are not cleared.

This feature is available only when the tamper is configured in level detection with filtering mode (TAMPFLT ≠ 00 and active mode is not selected). Refer to *Section : Level detection with filtering on tamper inputs (passive mode)*.

Timestamp on tamper event

With TAMPTS set to 1 in the RTC_CR, any internal or external tamper event causes a timestamp to occur. In case a timestamp occurs due to tamper event, either the TSF bit or the TSOVF bit is set in RTC_SR, in the same manner as if a normal timestamp event occurs.

Note: TSF is set up to 3 ck_apre cycles after TAMPxF flags. TSF is not set if RTCCLK is stopped (it is set when RTCCLK restarts).

Note: If TAMPxF is cleared before the expected rise of TSF, TSF is not set. Consequently, in case TAMPTS = 1, the software should either wait for timestamp flag before clearing the tamper flag, or should read the RTC counters values in the TAMP interrupt routine.

Edge detection on tamper inputs (passive mode)

If the TAMPFLT bits are 00, the TAMP_INx pins generate tamper detection events when either a rising edge or a falling edge is observed depending on the corresponding

TAMPxTRG bit. The internal pull-up resistors on the TAMP_INx inputs are deactivated when edge detection is selected.

Caution: When TAMPFLT = 00 and TAMPxTRG = 0 (rising edge detection), a tamper event may be detected by hardware if the tamper input is already at high level before enabling the tamper detection.

After a tamper event has been detected and cleared, the TAMP_INx should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (TAMP_BKPxR). This prevents the application from writing to the backup registers while the TAMP_INx input value still indicates a tamper detection. This is equivalent to a level detection on the TAMP_INx input.

Level detection with filtering on tamper inputs (passive mode)

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The TAMP_INx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the TAMP_INx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection. The TAMP_IN I/O schmitt trigger is enabled only during the precharge duration to avoid any extra consumption if the tamper switch is open (floating state).

Figure 273. Tamper sampling with precharge pulse

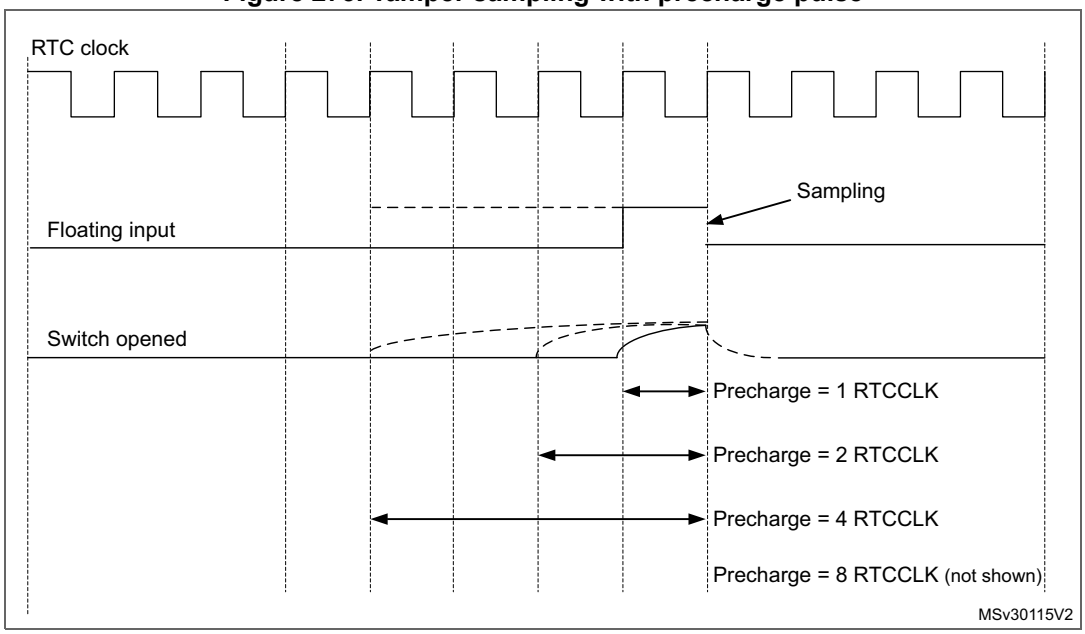
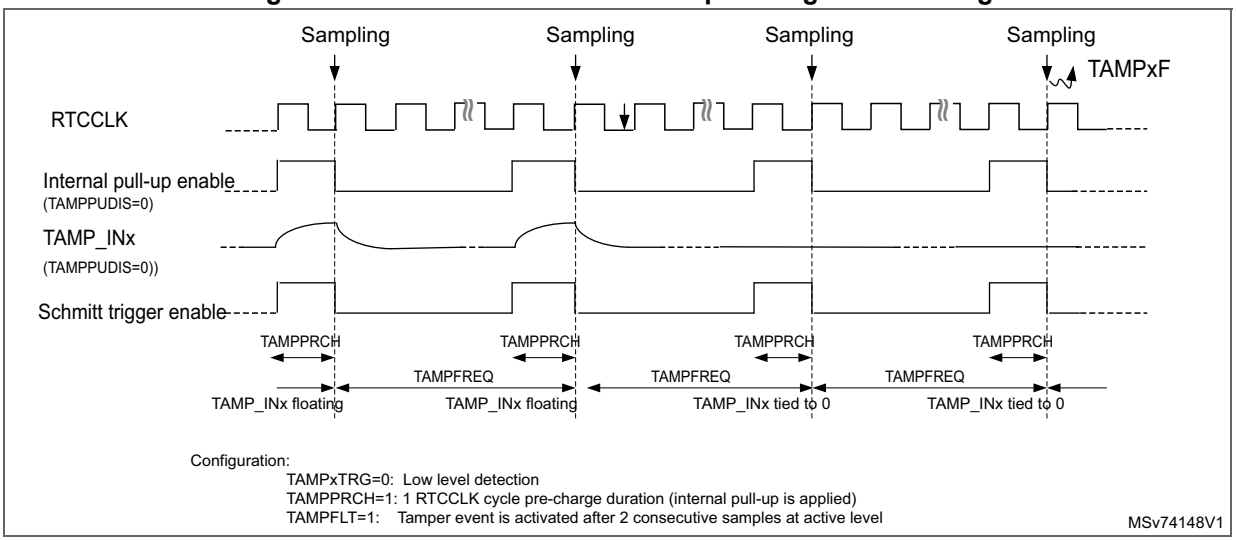


Figure 274. Low level detection with precharge and filtering



Note: Refer to the microcontroller datasheet for the electrical characteristics of the pull-up resistors.

Active tamper detection

When the TAMPxAM bit is set in the TAMP_ATCR, the tamper events are configured in active mode, which is based on a comparison between a TAMP_OUTy pin and a TAMP_INx pin. By default (ATOSHARE = 0) the comparison is made between TAMP_INx and TAMP_OUTx (y = x). When ATOSHARE bit is set, the same output can be used for several tamper inputs. The TAMP_OUTy function is enabled on the I/O as soon as it is selected for comparison with an active tamper input TAMP_INx (TAMPxEN = TAMPxAM = 1), thanks to ATOSHARE and ATOSELx bits. Refer to ATOSHARE and ATOSEL bits descriptions in the TAMP_ATCRx (x = 1, 2) registers.

Every two CK_ATPER cycles (CK_ATPER = 2^{ATPER} × CK_ATPRE), TAMP_OUTy output pin provides a value provided by a pseudo random number generator (PRNG). After outputting this value, the TAMP_OUTy pin outputs its opposite value one CK_ATPER cycle after.

Table 316. Active tamper output change period

ATCKSEL[3:0]	CK_ATPRE frequency	ATPER[2:0]	Tamper output change (CK_ATPER) frequency	Tamper output change period(1) (ms)
0x0	f_{RTCCLK}	0x0	f_{RTCCLK}	0.030
		0x1	$f_{RTCCLK}/2$	0.061
		0x2	$f_{RTCCLK}/4$	0.122
		0x3	$f_{RTCCLK}/8$	0.244
		0x4	$f_{RTCCLK}/16$	0.488
		0x5	$f_{RTCCLK}/32$	0.977
		0x6	$f_{RTCCLK}/64$	1.953
		0x7	$f_{RTCCLK}/128$	3.906
…	…	…	…	…
0x7	$f_{RTCCLK}/128$	0x0	$f_{RTCCLK}/128$	3.906
		0x1	$f_{RTCCLK}/256$	7.8125
		0x2	$f_{RTCCLK}/512$	15.625
		0x3	$f_{RTCCLK}/1024$	31.250
		0x4	$f_{RTCCLK}/2048$	62.5
		0x5	$f_{RTCCLK}/4096$	125
		0x6	$f_{RTCCLK}/8192$	250
		0x7	$f_{RTCCLK}/16384$	500
0xB	$f_{RTCCLK}/2048$(2)	0x0	$f_{RTCCLK}/2048$	62.5
		0x1	$f_{RTCCLK}/4096$	125
		0x2	$f_{RTCCLK}/8192$	250
		0x3	$f_{RTCCLK}/16384$	500
		0x4	$f_{RTCCLK}/32768$	1000
		0x5	$f_{RTCCLK}/65536$	2000
		0x6	$f_{RTCCLK}/131072$	4000
		0x7	$f_{RTCCLK}/262144$	8000

1. Assuming f_{RTCCLK} = 32768 Hz.
2. This setting requires that (PREDIV_A+1) = 128 and (PREDIV_S+1) is a multiple of 16.

PRNG is consumed by the selected tamper outputs at a different frequency depending on the number of selected tamper outputs. The number of selected outputs depends on TAMPxAM, TAMPxE, ATOSEL and ATOSHARE.

- When only 1 output is selected: PRNG is consumed every 16 CK_ATPER periods.
- When 2 outputs are selected: PRNG is consumed every 8 CK_ATPER periods.
- When 3 or 4 outputs are selected: PRNG is consumed every 4 CK_ATPER periods.

- When 5 or more outputs are selected: PRNG is consumed every 2 CK_ATPER periods

The PRNG needs minimum 9 CK_ATPRE cycles to output a new value. Consequently the minimum ATPER values for correct functionality are provided in the table below:

Table 317. Minimum ATPER value

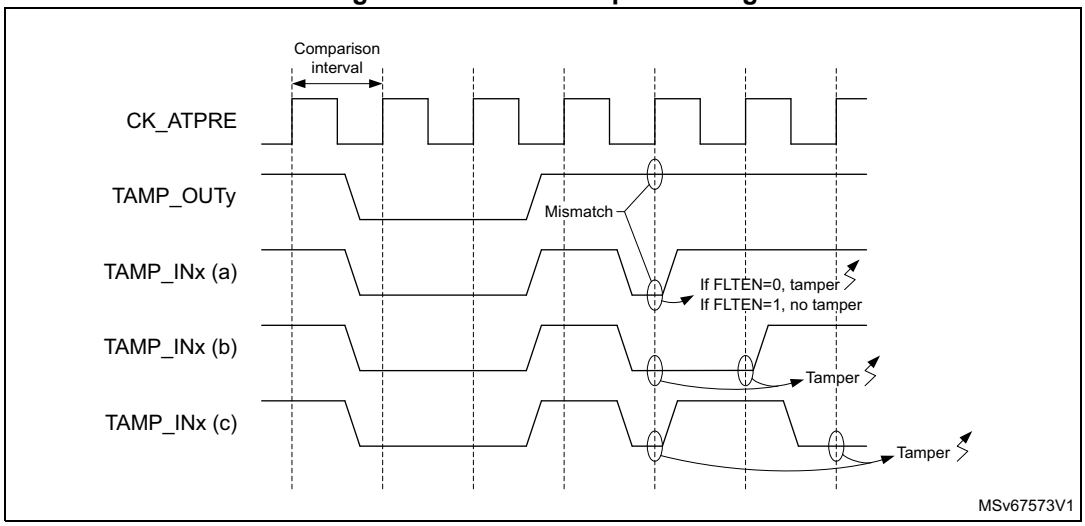
Number of selected outputs	Minimum ATPER
1	0
2	1
3 or 4	2
5 or more	3

The TAMP_INx pin is externally connected to TAMP_OUTy pin. The comparison is made between TAMP_OUTy output value and TAMP_INx received value, every CK_ATPRE cycle. In case a comparison mismatch occurs, the TAMPxF bit is set in the TAMP_SR register.

As an example, TAMP_OUT1 can be used for comparison with TAMP_IN1 and TAMP_IN2 by configuring and enabling both TAMP1 and TAMP2 in active mode, with ATOSHARE = 1, ATOSEL1 = 000 and ATOSEL2 = 000.

The active tamper can be combined with input filtering when FLTEN = 1. In this case, the tamper is detected only when 2 comparisons are false, in 4 consecutive comparison samples.

Figure 275. Active tamper filtering



As illustrated in *Figure 275*, if FLTEN = 0, any mismatch between the TAMP_OUTy output and the associated TAMP_INx input when the latter is sampled generates a tamper. This is the case in all three examples (a), (b) and (c).

If FLTEN = 1, example (a) does not generate a tamper, since only one mismatch is detected in four consecutive comparisons. In example (b), a tamper is generated since two successive mismatches are detected. Example (c) also generates a tamper, since two mismatches occur in four consecutive comparisons, even though the mismatches do not occur on successive samples.

Setting FLTEN = 1 avoids unwanted detection of tampers due to glitches, bounce or transitory states on the TAMP_INx inputs, by ignoring single pulses which are shorter than one period of CK_ATPRE, programmed in the ATCKSEL field of the TAMP_ATCR1 register. The minimum filtered pulse width is listed in *Table 318* for each possible setting of ATCKSEL, assuming f_{RTCCLK} = 32.768 kHz.

Table 318. Active tamper filtered pulse duration

ATCKSEL[3:0]	CK_ATPRE frequency	Minimum filtered pulse width (ms)
0x0	f_{RTCCLK}	0.030
0x1	$f_{RTCCLK}/2$	0.061
0x2	$f_{RTCCLK}/4$	0.122
0x3	$f_{RTCCLK}/8$	0.244
0x4	$f_{RTCCLK}/16$	0.488
0x5	$f_{RTCCLK}/32$	0.977
0x6	$f_{RTCCLK}/64$	1.953
0x7	$f_{RTCCLK}/128$	3.906
0xB	$f_{RTCCLK}/2048$	62.500(1)

1. This setting requires that (PREDIV_A+1) = 128 and (PREDIV_S+1) is a multiple of 16.

Note: Multiple pulses which are shorter than one CK_ATPRE period may nevertheless cause a tamper if they result in two mismatches in four consecutive comparisons.

Caution: Entering RTC initialization mode stops CK_ATPRE and CK_ATPER clocks when ATCKSEL[3] = 1. Therefore, TAMP_OUTy pin stops toggling until INIT mode exit.

Refer to section *Section : Calendar initialization and configuration*.

Refer also to *RTC alarm A subsecond register (RTC_ALRMASSR)*, *RTC alarm B subsecond register (RTC_ALRMBSSR)*, *RTC alarm A binary mode register (RTC_ALRABINR)* and *RTC alarm B binary mode register (RTC_ALRBBINR)* in case RTC binary mode is used in conjunction with ATCKSEL[3] = 1.

Caution: The active tamper detection is no more functional in case of calendar overflow when ATCKSEL[3] = 1. It is mandatory to enable the internal tamper 5 on calendar overflow to ensure tamper protection.

The pseudo-random generator must be initialized with a seed. This is done by writing consecutively four 32-bit random values in the TAMP_ATSEEDR register. Programming the seed automatically sends it to the PRNG. As long as the new seed is transferred and elaborated by the PRNG, the SEEDF bit is set in the TAMP_ATOR and it is not allowed to switch off the TAMP APB clock. The duration of the elaboration is up to 184 APB clock cycles after the forth seed is written. Consequently, after writing a new seed, the user must wait until SEEDF is cleared before entering low-power modes.

The active tamper outputs are activated only after the first seed is written and the elaboration is completed. Then new seeds can be written and elaborated during active tamper activity.

Active tamper initialization

Here is the software procedure to initialize the active tampers after system reset:

Read INITS in TAMP_ATOR register.

- If INITS = 0x0 (initialization was not done):
  a) Write TAMP_ATCR to configure Active tamper clock, filter and output sharing if any, and active mode.
  b) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  c) Write SEED by writing four times in the TAMP_ATSEEDR.
  d) Wait until SEEDF = 0 in TAMP_ATOR. Backup registers are then protected by active tamper.
- If INITS = 0x1 (initialization already done):
  No initialization. To increase randomness a new SEED should be provided regularly. When a new SEED is provided, wait until SEEDF = 0 before entering a low-power mode which switches off the TAMP APB clock.
- In case the tampers are disabled by software, and re-enabled afterwards, the SEED must be written after enabling tampers:
  a) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  b) Write SEED by writing four times in the TAMP_ATSEEDR.
  c) Wait until SEEDF = 0 in TAMP_ATOR. Backup registers are then protected by active tamper.

34.4 TAMP low-power modes

Table 319. Effect of low-power modes on TAMP

Mode	Description
Sleep	No effect. TAMP interrupts cause the device to exit the Sleep mode.
Stop 0, 1, and 2	No effect on all features, except for level detection with filtering and active tamper modes which remain active only when the clock source is LSE or LSI. TAMP interrupts cause the device to exit the Stop mode.
Stop 3	Only passive tamper events cause the device to exit the Stop mode.
Standby retention	No effect on all features when retained with RTCSB in PWR_CR1, except for level detection with filtering and active tamper modes which remain active only when the clock source is LSE or LSI. TAMP interrupts cause the device to exit the Standby retention mode.
Standby	The TAMP is powered-down and must be reinitialized after exiting the Standby mode.

Table 320. TAMP pins functionality over modes

Pin name	Functional in all low-power modes except Standby
TAMP_IN[4:1]	Yes
TAMP_OUT[4:1]	Yes(1)

1. Only functional in Stop 0, 1, 2, and Standby retention.

34.5 TAMP interrupts

The interrupt channel is set in the masked interrupt status register or in the secure masked interrupt status register depending on its security mode configuration (TAMPSEC).

Table 321. Interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit	Interrupt clear method	Exit from low-power modes
TAMP	Tamper x(2)	TAMPxF	TAMPxIE	Write 1 in CTAMPxF	Yes(3)
	Internal tamper y(2)	ITAMPyF	ITAMPyIE	Write 1 in CITAMPyF	Yes(3)

1. The event flags are in the TAMP_SR register.
2. The number of tampers and internal tampers events depend on products.
3. Refer to *Table 319: Effect of low-power modes on TAMP* for more details about available features in the low-power modes.

34.6 TAMP registers

Refer to *Section 1.2* of the reference manual for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by words (32-bit).

34.6.1 TAMP control register 1 (TAMP_CR1)

This register can be protected against nonsecure access. Refer to *Section 34.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 34.3.7: TAMP privilege protection modes*.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ITAMP13E	ITAMP12E	ITAMP11E	Res.	ITAMP9E	ITAMP8E	ITAMP7E	ITAMP6E	ITAMP5E	Res.	ITAMP3E	Res.	Res.
			rw	rw	rw		rw	rw	rw	rw	rw		rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP4E	TAMP3E	TAMP2E	TAMP1E
												rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **ITAMP13E**: Internal tamper 13 enable
0: Internal tamper 13 disabled.
1: Internal tamper 13 enabled.

Bit 27 **ITAMP12E**: Internal tamper 12 enable
0: Internal tamper 12 disabled.
1: Internal tamper 12 enabled.

Bit 26 **ITAMP11E**: Internal tamper 11 enable
0: Internal tamper 11 disabled.
1: Internal tamper 11 enabled.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9E**: Internal tamper 9 enable
0: Internal tamper 9 disabled.
1: Internal tamper 9 enabled.

Bit 23 **ITAMP8E**: Internal tamper 8 enable
0: Internal tamper 8 disabled.
1: Internal tamper 8 enabled.

Bit 22 **ITAMP7E**: Internal tamper 7 enable
0: Internal tamper 7 disabled.
1: Internal tamper 7 enabled

Bit 21 **ITAMP6E**: Internal tamper 6 enable
0: Internal tamper 6 disabled.
1: Internal tamper 6 enabled.

Bit 20 **ITAMP5E**: Internal tamper 5 enable
0: Internal tamper 5 disabled.
1: Internal tamper 5 enabled.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3E**: Internal tamper 3 enable
0: Internal tamper 3 disabled.
1: Internal tamper 3 enabled.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **TAMP4E**: Tamper detection on TAMP_IN4 enable(1)
0: Tamper detection on TAMP_IN4 is disabled.
1: Tamper detection on TAMP_IN4 is enabled.

Bit 2 **TAMP3E**: Tamper detection on TAMP_IN3 enable(1)
0: Tamper detection on TAMP_IN3 is disabled.
1: Tamper detection on TAMP_IN3 is enabled.

Bit 1 **TAMP2E**: Tamper detection on TAMP_IN2 enable(1)
0: Tamper detection on TAMP_IN2 is disabled.
1: Tamper detection on TAMP_IN2 is enabled.

Bit 0 **TAMP1E**: Tamper detection on TAMP_IN1 enable(1)
0: Tamper detection on TAMP_IN1 is disabled.
1: Tamper detection on TAMP_IN1 is enabled.

1. Tamper detection mode (selected with TAMP_FLTCR, TAMP_ATCR1, TAMP_ATCR2 registers and TAMPxTRG bits in TAMP_CR2), must be configured before enabling the tamper detection.

34.6.2 TAMP control register 2 (TAMP_CR2)

This register can be protected against nonsecure access. Refer to *Section 34.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 34.3.7: TAMP privilege protection modes*.

Address offset: 0x04

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TAMP4 TRG	TAMP3 TRG	TAMP2 TRG	TAMP1 TRG	BK ERASE	BK BLOCK	Res.	Res.	Res.	TAMP3 MSK	TAMP2 MSK	TAMP1 MSK
				rw	rw	rw	rw	w	rw				rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP4 POM	TAMP3 POM	TAMP2 POM	TAMP1 POM
												rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 **TAMP4TRG**: Active level for tamper 4 input (active mode disabled)
0: If TAMPFLT ≠ 00 tamper 4 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 4 input rising edge triggers a tamper detection event.
1: If TAMPFLT ≠ 00 tamper 4 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 4 input falling edge triggers a tamper detection event.

Bit 26 **TAMP3TRG**: Active level for tamper 3 input
0: If TAMPFLT ≠ 00 tamper 3 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 3 input rising edge triggers a tamper detection event.
1: If TAMPFLT ≠ 00 tamper 3 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 3 input falling edge triggers a tamper detection event.

Bit 25 **TAMP2TRG**: Active level for tamper 2 input

0: If TAMPFLT ≠ 00 tamper 2 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 2 input rising edge triggers a tamper detection event.

1: If TAMPFLT ≠ 00 tamper 2 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 2 input falling edge triggers a tamper detection event.

Bit 24 **TAMP1TRG**: Active level for tamper 1 input

0: If TAMPFLT ≠ 00 tamper 1 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 1 input rising edge triggers a tamper detection event.

1: If TAMPFLT ≠ 00 tamper 1 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 1 input falling edge triggers a tamper detection event.

Bit 23 **BKERASE**: Backup registers and device secrets(1) erase

Writing '1' to this bit reset the backup registers and device secrets(1). Writing 0 has no effect. This bit is always read as 0.

Bit 22 **BKBLOCK**: Backup registers and device secrets(1) access blocked

0: backup registers and device secrets(1) can be accessed if no tamper flag is set

1: backup registers and device secrets(1) cannot be accessed

Bits 21:19 Reserved, must be kept at reset value.

Bit 18 **TAMP3MSK**: Tamper 3 mask

0: Tamper 3 event generates a trigger event and TAMP3F must be cleared by software to allow next tamper event detection.

1: Tamper 3 event generates a trigger event. TAMP3F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.

The tamper 3 interrupt must not be enabled when TAMP3MSK is set.

Bit 17 **TAMP2MSK**: Tamper 2 mask

0: Tamper 2 event generates a trigger event and TAMP2F must be cleared by software to allow next tamper event detection.

1: Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.

The tamper 2 interrupt must not be enabled when TAMP2MSK is set.

Bit 16 **TAMP1MSK**: Tamper 1 mask

0: Tamper 1 event generates a trigger event and TAMP1F must be cleared by software to allow next tamper event detection.

1: Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.

The tamper 1 interrupt must not be enabled when TAMP1MSK is set.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **TAMP4POM**: Tamper 4 potential mode

0: Tamper 4 event detection is in confirmed mode(1).

1: Tamper 4 event detection is in potential mode(2).

Bit 2 **TAMP3POM**: Tamper 3 potential mode
0: Tamper 3 event detection is in confirmed mode(1).
1: Tamper 3 event detection is in potential mode(2).

Bit 1 **TAMP2POM**: Tamper 2 potential mode
0: Tamper 2 event detection is in confirmed mode(1).
1: Tamper 2 event detection is in potential mode(2).

Bit 0 **TAMP1POM**: Tamper 1 potential mode
0: Tamper 1 event detection is in confirmed mode(1).
1: Tamper 1 event detection is in potential mode(2).

1. The effects of tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in *Table 314: TAMP interconnection*.
2. The effects of tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in *Table 314: TAMP interconnection*.

34.6.3 TAMP control register 3 (TAMP_CR3)

This register can be protected against nonsecure access. Refer to *Section 34.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 34.3.7: TAMP privilege protection modes*.

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ITAMP 13POM	ITAMP 12POM	ITAMP 11POM	Res.	ITAMP9 POM	ITAMP8 POM	ITAMP7 POM	ITAMP6 POM	ITAMP5 POM	Res.	ITAMP3 POM	Res.	Res.
			rw	rw	rw		rw	rw	rw	rw	rw		rw		

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ITAMP13POM**: Internal tamper 13 potential mode
0: Internal tamper 13 event detection is in confirmed mode(1).
1: Internal tamper 13 event detection is in potential mode(2).

Bit 11 **ITAMP12POM**: Internal tamper 12 potential mode
0: Internal tamper 12 event detection is in confirmed mode(1).
1: Internal tamper 12 event detection is in potential mode(2).

Bit 10 **ITAMP11POM**: Internal tamper 11 potential mode
0: Internal tamper 11 event detection is in confirmed mode(1).
1: Internal tamper 11 event detection is in potential mode(2).

Bit 9 Reserved, must be kept at reset value.

Bit 8 **ITAMP9POM**: Internal tamper 9 potential mode
0: Internal tamper 9 event detection is in confirmed mode(1).
1: Internal tamper 9 event detection is in potential mode(2).

Bit 7 **ITAMP8POM**: Internal tamper 8 potential mode
0: Internal tamper 8 event detection is in confirmed mode(1).
1: Internal tamper 8 event detection is in potential mode(2).

Bit 6 **ITAMP7POM**: Internal tamper 7 potential mode
0: Internal tamper 7 event detection is in confirmed mode(1).
1: Internal tamper 7 event detection is in potential mode(2).

Bit 5 **ITAMP6POM**: Internal tamper 6 potential mode
0: Internal tamper 6 event detection is in confirmed mode(1).
1: Internal tamper 6 event detection is in potential mode(2).

Bit 4 **ITAMP5POM**: Internal tamper 5 potential mode
0: Internal tamper 5 event detection is in confirmed mode(1).
1: Internal tamper 5 event detection is in potential mode(2).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **ITAMP3POM**: Internal tamper 3 potential mode
0: Internal tamper 3 event detection is in confirmed mode(1).
1: Internal tamper 3 event detection is in potential mode(2).

Bit 1 Reserved, must be kept at reset value.

Bit 0 Reserved, must be kept at reset value.

1. The effects of internal tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in *Table 314: TAMP interconnection*
2. The effects of internal tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in *Table 314: TAMP interconnection*.

34.6.4 TAMP filter control register (TAMP_FLTCR)

This register can be protected against nonsecure access. Refer to *Section 34.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 34.3.7: TAMP privilege protection modes*.

Address offset: 0x0C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP PUDIS	TAMPPRCH [1:0]		TAMPFLT [1:0]		TAMPFREQ [2:0]		
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **TAMPPUDIS**: TAMP_INx pull-up disable

This bit determines if each of the TAMPx pins are precharged before each sample.

0: Precharge TAMP_INx pins before sampling (enable internal pull-up)

1: Disable precharge of TAMP_INx pins.

Bits 6:5 **TAMPPRCH[1:0]**: TAMP_INx precharge duration

These bit determines the duration of time during which the pull-up/is activated before each sample. TAMPPRCH is valid for each of the TAMP_INx inputs.

0x0: 1 RTCCLK cycle

0x1: 2 RTCCLK cycles

0x2: 4 RTCCLK cycles

0x3: 8 RTCCLK cycles

Bits 4:3 **TAMPFLT[1:0]**: TAMP_INx filter count

These bits determines the number of consecutive samples at the specified level (TAMP*TRG) needed to activate a tamper event. TAMPFLT is valid for each of the TAMP_INx inputs.

0x0: Tamper event is activated on edge of TAMP_INx input transitions to the active level (no internal pull-up on TAMP_INx input).

0x1: Tamper event is activated after 2 consecutive samples at the active level.

0x2: Tamper event is activated after 4 consecutive samples at the active level.

0x3: Tamper event is activated after 8 consecutive samples at the active level.

Bits 2:0 **TAMPFREQ[2:0]**: Tamper sampling frequency

Determines the frequency at which each of the TAMP_INx inputs are sampled.

0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)

0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)

0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)

0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)

0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)

0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)

0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)

0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

Note: This register concerns only the tamper inputs in passive mode.

34.6.5 TAMP active tamper control register 1 (TAMP_ATCR1)

This register can be protected against nonsecure access. Refer to *Section 34.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 34.3.7: TAMP privilege protection modes*.

Address offset: 0x10

Backup domain reset value: 0x0007 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTEN	ATO SHARE	Res.	Res.	Res.	ATPER[2:0]			Res.	Res.	Res.	Res.	ATCKSEL[3:0]			
rw	rw				rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATOSEL4[1:0]		ATOSEL3[1:0]		ATOSEL2[1:0]		ATOSEL1[1:0]		Res.	Res.	Res.	Res.	TAMP4 AM	TAMP3 AM	TAMP2 AM	TAMP1 AM
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bit 31 **FLTEN**: Active tamper filter enable

0: Active tamper filtering disable

1: Active tamper filtering enable: a tamper event is detected when 2 comparison mismatches occur out of 4 consecutive samples.

Bit 30 **ATOSHARE**: Active tamper output sharing

0: Each active tamper input TAMP_INi is compared with its dedicated output TAMP_OUTi

1: Each active tamper input TAMP_INi is compared with TAMPOUTSELi defined by ATOSELi bits.

Bits 29:27 Reserved, must be kept at reset value.

Bits 26:24 **ATPER[2:0]**: Active tamper output change period

The tamper output is changed every CK_ATPER = (2^{ATPER} x CK_ATPRE) cycles. Refer to *Table 317: Minimum ATPER value*.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ATCKSEL[3:0]**: Active tamper RTC asynchronous prescaler clock selection

These bits selects the RTC asynchronous prescaler stage output. The selected clock is CK_ATPRE.

0000: RTCCLK is selected
0001: RTCCLK/2 is selected
0010: RTCCLK/4 is selected
0011: RTCCLK/8 is selected
0100: RTCCLK/16 is selected
0101: RTCCLK/32 is selected
0110: RTCCLK/64 is selected
0111: RTCCLK/128 is selected
1011: RTCCLK/2048 is selected when (PREDIV_A+1) = 128 and (PREDIV_S+1) is a multiple of 16.
Others: Reserved

Note: These bits can be written only when all active tampers are disabled. The write protection remains for up to 1.5 CK_ATPRE cycles after all the active tampers are disable.

Bits 15:14 **ATOSEL4[1:0]**: Active tamper shared output 4 selection
00: TAMPOUTSEL4 = TAMP_OUT1
01: TAMPOUTSEL4 = TAMP_OUT2
10: TAMPOUTSEL4 = TAMP_OUT3
11: TAMPOUTSEL4 = TAMP_OUT4
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 13:12 **ATOSEL3[1:0]**: Active tamper shared output 3 selection
00: TAMPOUTSEL3 = TAMP_OUT1
01: TAMPOUTSEL3 = TAMP_OUT2
10: TAMPOUTSEL3 = TAMP_OUT3
11: TAMPOUTSEL3 = TAMP_OUT4
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 11:10 **ATOSEL2[1:0]**: Active tamper shared output 2 selection
00: TAMPOUTSEL2 = TAMP_OUT1
01: TAMPOUTSEL2 = TAMP_OUT2
10: TAMPOUTSEL2 = TAMP_OUT3
11: TAMPOUTSEL2 = TAMP_OUT4
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 9:8 **ATOSEL1[1:0]**: Active tamper shared output 1 selection
00: TAMPOUTSEL1 = TAMP_OUT1
01: TAMPOUTSEL1 = TAMP_OUT2
10: TAMPOUTSEL1 = TAMP_OUT3
11: TAMPOUTSEL1 = TAMP_OUT4
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **TAMP4AM**: Tamper 4 active mode
0: Tamper 4 detection mode is passive.
1: Tamper 4 detection mode is active.

Bit 2 **TAMP3AM**: Tamper 3 active mode
0: Tamper 3 detection mode is passive.
1: Tamper 3 detection mode is active.

Bit 1 **TAMP2AM**: Tamper 2 active mode
0: Tamper 2 detection mode is passive.
1: Tamper 2 detection mode is active.

Bit 0 **TAMP1AM**: Tamper 1 active mode
0: Tamper 1 detection mode is passive.
1: Tamper 1 detection mode is active.

Note: Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register.

All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write).

All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write).

A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.

34.6.6 TAMP active tamper seed register (TAMP_ATSEEDR)

This register can be protected against nonsecure access. Refer to *Section 34.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 34.3.7: TAMP privilege protection modes*.

Address offset: 0x14

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 **SEED[31:0]**: Pseudo-random generator seed value

This register must be written four times with 32-bit values to provide the 128-bit seed to the PRNG. Writing to this register automatically sends the seed value to the PRNG.

34.6.7 TAMP active tamper output register (TAMP_ATOR)

This register can be protected against nonsecure access. Refer to *Section 34.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 34.3.7: TAMP privilege protection modes*.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected, except for SEEDF which is reset to 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INITS	SEEDF	Res.	Res.	Res.	Res.	Res.	Res.	PRNG[7:0]							
r	r							r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **INITS**: Active tamper initialization status

This flag is set by hardware when the PRNG has absorbed the first 128-bit seed, meaning that the enabled active tampers are functional. This flag is cleared when the active tampers are disabled.

Bit 14 **SEEDF**: Seed running flag

This flag is set by hardware when a new seed is written in the TAMP_ATSEEDR. It is cleared by hardware when the PRNG has absorbed this new seed, and by system reset. The TAMP APB clock must not be switched off as long as SEEDF is set.

Bits 13:8 Reserved, must be kept at reset value.

Bits 7:0 **PRNG[7:0]**: Pseudo-random generator value

This field provides the values of the PRNG output. Because of potential inconsistencies due to synchronization delays, PRNG must be read at least twice. The read value is correct if it is equal to previous read value.

This field can only be read when the APB is in secure mode.

34.6.8 TAMP active tamper control register 2 (TAMP_ATCR2)

This register can be protected against nonsecure access. Refer to *Section 34.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 34.3.7: TAMP privilege protection modes*.

Address offset: 0x1C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATOSEL4[2:0]			ATOSEL3[2]
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATOSEL3[1:0]		ATOSEL2[2:0]			ATOSEL1[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:17 **ATOSEL4[2:0]**: Active tamper shared output 4 selection

000: TAMPOUTSEL4 = TAMP_OUT1
001: TAMPOUTSEL4 = TAMP_OUT2
010: TAMPOUTSEL4 = TAMP_OUT3
011: TAMPOUTSEL4 = TAMP_OUT4
100: TAMPOUTSEL4 = TAMP_OUT5
101: TAMPOUTSEL4 = TAMP_OUT6
110: TAMPOUTSEL4 = TAMP_OUT7
111: TAMPOUTSEL4 = TAMP_OUT8

If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 18:17 are the mirror of ATOSEL4[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 16:14 **ATOSEL3[2:0]**: Active tamper shared output 3 selection

000: TAMPOUTSEL3 = TAMP_OUT1
001: TAMPOUTSEL3 = TAMP_OUT2
010: TAMPOUTSEL3 = TAMP_OUT3
011: TAMPOUTSEL3 = TAMP_OUT4
100: TAMPOUTSEL3 = TAMP_OUT5
101: TAMPOUTSEL3 = TAMP_OUT6
110: TAMPOUTSEL3 = TAMP_OUT7
111: TAMPOUTSEL3 = TAMP_OUT8

If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 15:14 are the mirror of ATOSEL3[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 13:11 **ATOSEL2[2:0]**: Active tamper shared output 2 selection
000: TAMPOUTSEL2 = TAMP_OUT1
001: TAMPOUTSEL2 = TAMP_OUT2
010: TAMPOUTSEL2 = TAMP_OUT3
011: TAMPOUTSEL2 = TAMP_OUT4
100: TAMPOUTSEL2 = TAMP_OUT5
101: TAMPOUTSEL2 = TAMP_OUT6
110: TAMPOUTSEL2 = TAMP_OUT7
111: TAMPOUTSEL2 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.
Bits 12:11 are the mirror of ATOSEL2[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 10:8 **ATOSEL1[2:0]**: Active tamper shared output 1 selection
000: TAMPOUTSEL1 = TAMP_OUT1
001: TAMPOUTSEL1 = TAMP_OUT2
010: TAMPOUTSEL1 = TAMP_OUT3
011: TAMPOUTSEL1 = TAMP_OUT4
100: TAMPOUTSEL1 = TAMP_OUT5
101: TAMPOUTSEL1 = TAMP_OUT6
110: TAMPOUTSEL1 = TAMP_OUT7
111: TAMPOUTSEL1 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.
Bits 9:8 are the mirror of ATOSEL1[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 7:0 Reserved, must be kept at reset value.

Note: Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register.

All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write).

All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write).

A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.

34.6.9 TAMP secure configuration register (TAMP_SECCFGR)

If TZEN = 1, this register can be written only when the APB access is secure. If TZEN = 0, BKPRWSEC[7:0], BKPWSEC[7:0] and BHKLOCK can be written with nonsecure APB access, and TAMPSEC, CNT1SEC cannot be written.

This register can be globally write-protected, or each bit of this register can be individually write-protected against unprivileged access depending on the TAMP_PRIVCFGR configuration (refer to *Section 34.3.7: TAMP privilege protection modes*).

Address offset: 0x20

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TAMPSEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPWSEC[7:0]							
rw								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT1SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRWSEC[7:0]							
rw								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TAMPSEC**: Tamper protection (excluding monotonic counters and backup registers)

0: Tamper configuration and interrupt can be written when the APB access is secure or nonsecure.

1: Tamper configuration and interrupt can be written only when the APB access is secure.

Note: Refer to Section 34.3.5: TAMP secure protection modes for details on the read protection.

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 Reserved, must be kept at reset value.

Bits 23:16 **BKPWSEC[7:0]**: Backup registers write protection offset

BKPWSEC value must be from 0 to 32.

Protection zone 2 is defined for backup registers from TAMP_BKPyR (y = BKPRWSEC) to TAMP_BKPzR (z = BKPWSEC-1, with BKPWSEC > BKPRWSEC):

– if TZEN=1, these backup registers can be written only with secure access.

– They can be read with secure or nonsecure access.

If BKPWSEC = 0 or if BKPWSEC ≤ BKPRWSEC: there is no protection zone 2.

Protection zone 3 is defined for backup registers from TAMP_BKPtR (t = BKPWSEC if BKPWSEC ≥ BKPRWSEC, else t = BKPRWSEC).

– They can be read or written with secure or nonsecure access.

If BKPWSEC = 32: there is no protection zone 3.

Refer to *Figure 272: Backup registers protection zones*.

Note: If TZEN=0: the protection zone 2 can be read and written with nonsecure access.

Note: If BKPWPRIV is set, BKPWSEC[7:0] can be written only in privileged mode.

Bit 15 **CNT1SEC**: Monotonic counter 1 secure protection
0: Monotonic counter 1 (TAMP_COUNT1R) can be read and written when the APB access is secure or nonsecure.
1: Monotonic counter 1 (TAMP_COUNT1R) can be read and written only when the APB access is secure.

Bits 14:8 Reserved, must be kept at reset value.

Bits 7:0 **BKPRWSEC[7:0]**: Backup registers read/write protection offset
BKPRWSEC value must be from 0 to 32.

Protection zone 1 is defined for backup registers from TAMP_BKP0R to TAMP_BKPxR (x = BKPRWSEC-1, with BKPRWSEC ≥ 1).
– if TZEN=1, these backup registers can be read and written only with secure access.
If BKPRWSEC = 0: there is no protection zone 1.

Refer to *Figure 272: Backup registers protection zones*.

Note: If TZEN=0: the protection zone 1 can be read and written with nonsecure access.

Note: If BKPRWPRIV or BKPWPRIV is set, BKPRWSEC[7:0] can be written only in privileged mode.

34.6.10 TAMP privilege configuration register (TAMP_PRIVCFGR)

This register can be written only when the APB access is privileged.

When TZEN = 1, this register can be write-protected, or each bit of this register can be individually write-protected against nonsecure access depending on the TAMP_SECCFGR configuration (refer to *Section 34.3.5: TAMP secure protection modes*).

Address offset: 0x24

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TAMP PRIV	BKP WPRIV	BKPR WPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT1 PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

Bit 31 **TAMPPRIV**: Tamper privilege protection (excluding monotonic counters and backup registers)
0: Tamper configuration and interrupt can be written with privileged or unprivileged access.
1: Tamper configuration and interrupt can be written only with privileged access.
Note: Refer to Section 34.3.7: TAMP privilege protection modes for details on the read protection.

Bit 30 **BKPWPRIV**: Backup registers zone 2 privilege protection
0: Backup registers zone 2 can be written with privileged or unprivileged access.
1: Backup registers zone 2 can be written only with privileged access.

Bit 29 **BKPRWPRIV**: Backup registers zone 1 privilege protection
0: Backup registers zone 1 can be read and written with privileged or unprivileged access.
1: Backup registers zone 1 can be read and written only with privileged access

Bits 28:16 Reserved, must be kept at reset value.

Bit 15 **CNT1PRIV**: Monotonic counter 1 privilege protection
0: Monotonic counter 1 (TAMP_COUNT1R) can be read and written when the APB access is privileged or unprivileged.
1: Monotonic counter 1 (TAMP_COUNT1R) can be read and written only when the APB access is privileged.

Bits 14:0 Reserved, must be kept at reset value.

34.6.11 TAMP interrupt enable register (TAMP_IER)

This register can be protected against nonsecure access. Refer to *Section 34.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 34.3.7: TAMP privilege protection modes*.

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ITAMP13IE	ITAMP12IE	ITAMP11IE	Res.	ITAMP9IE	ITAMP8IE	ITAMP7IE	ITAMP6IE	ITAMP5IE	Res.	ITAMP3IE	Res.	Res.
			rw	rw	rw		rw	rw	rw	rw	rw		rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP4IE	TAMP3IE	TAMP2IE	TAMP1IE
												rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **ITAMP13IE**: Internal tamper 13 interrupt enable
0: Internal tamper 13 interrupt disabled.
1: Internal tamper 13 interrupt enabled.

Bit 27 **ITAMP12IE**: Internal tamper 12 interrupt enable
0: Internal tamper 12 interrupt disabled.
1: Internal tamper 12 interrupt enabled.

Bit 26 **ITAMP11IE**: Internal tamper 11 interrupt enable
0: Internal tamper 11 interrupt disabled.
1: Internal tamper 11 interrupt enabled.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9IE**: Internal tamper 9 interrupt enable
0: Internal tamper 9 interrupt disabled.
1: Internal tamper 9 interrupt enabled.

Bit 23 **ITAMP8IE**: Internal tamper 8 interrupt enable
0: Internal tamper 8 interrupt disabled.
1: Internal tamper 8 interrupt enabled.

Bit 22 **ITAMP7IE**: Internal tamper 7 interrupt enable
0: Internal tamper 7 interrupt disabled.
1: Internal tamper 7 interrupt enabled.

Bit 21 **ITAMP6IE**: Internal tamper 6 interrupt enable
0: Internal tamper 6 interrupt disabled.
1: Internal tamper 6 interrupt enabled.

Bit 20 **ITAMP5IE**: Internal tamper 5 interrupt enable
0: Internal tamper 5 interrupt disabled.
1: Internal tamper 5 interrupt enabled.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3IE**: Internal tamper 3 interrupt enable
0: Internal tamper 3 interrupt disabled.
1: Internal tamper 3 interrupt enabled.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **TAMP4IE**: Tamper 4 interrupt enable
0: Tamper 4 interrupt disabled.
1: Tamper 4 interrupt enabled.

Bit 2 **TAMP3IE**: Tamper 3 interrupt enable
0: Tamper 3 interrupt disabled.
1: Tamper 3 interrupt enabled..

Bit 1 **TAMP2IE**: Tamper 2 interrupt enable
0: Tamper 2 interrupt disabled.
1: Tamper 2 interrupt enabled.

Bit 0 **TAMP1IE**: Tamper 1 interrupt enable
0: Tamper 1 interrupt disabled.
1: Tamper 1 interrupt enabled.

34.6.12 TAMP status register (TAMP_SR)

This register can be protected against nonsecure access. Refer to *Section 34.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 34.3.7: TAMP privilege protection modes*.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ITAMP13F	ITAMP12F	ITAMP11F	Res.	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6F	ITAMP5F	Res.	ITAMP3F	Res.	Res.
			r	r	r		r	r	r	r	r		r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP4F	TAMP3F	TAMP2F	TAMP1F
												r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **ITAMP13F**: Internal tamper 13 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 13.

Bit 27 **ITAMP12F**: Internal tamper 12 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 12.

Bit 26 **ITAMP11F**: Internal tamper 11 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 11.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9F**: Internal tamper 9 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 9.

Bit 23 **ITAMP8F**: Internal tamper 8 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 8.

Bit 22 **ITAMP7F**: Internal tamper 7 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 7.

Bit 21 **ITAMP6F**: Internal tamper 6 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 6.

Bit 20 **ITAMP5F**: Internal tamper 5 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 5.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3F**: Internal tamper 3 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 3.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **TAMP4F**: TAMP4 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP4 input.

Bit 2 **TAMP3F**: TAMP3 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP3 input.

Bit 1 **TAMP2F**: TAMP2 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP2 input.

Bit 0 **TAMP1F**: TAMP1 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP1 input.

34.6.13 TAMP nonsecure masked interrupt status register (TAMP_MISR)

This register can be protected against unprivileged access. Refer to *Section 34.3.7: TAMP privilege protection modes*.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	Res.	ITAMP3MF	Res.	Res.
			r	r	r		r	r	r	r	r		r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
												r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **ITAMP13MF**: internal tamper 13 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 13 nonsecure interrupt is raised.

Bit 27 **ITAMP12MF**: internal tamper 12 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 12 nonsecure interrupt is raised.

Bit 26 **ITAMP11MF**: internal tamper 11 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 11 nonsecure interrupt is raised.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9MF**: internal tamper 9 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 9 nonsecure interrupt is raised.

Bit 23 **ITAMP8MF**: Internal tamper 8 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 8 nonsecure interrupt is raised.

Bit 22 **ITAMP7MF**: Internal tamper 7 tamper nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 7 nonsecure interrupt is raised.

Bit 21 **ITAMP6MF**: Internal tamper 6 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 6 nonsecure interrupt is raised.

Bit 20 **ITAMP5MF**: Internal tamper 5 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 5 nonsecure interrupt is raised.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3MF**: Internal tamper 3 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 3 nonsecure interrupt is raised.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **TAMP4MF**: TAMP4 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 4 nonsecure interrupt is raised.

Bit 2 **TAMP3MF**: TAMP3 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 3 nonsecure interrupt is raised.

Bit 1 **TAMP2MF**: TAMP2 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 2 nonsecure interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 1 nonsecure interrupt is raised.

34.6.14 TAMP secure masked interrupt status register (TAMP_SMISR)

This register can be protected against nonsecure access. Refer to *Section 34.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 34.3.7: TAMP privilege protection modes*.

Address offset: 0x38

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	Res.	ITAMP3MF	Res.	Res.
			r	r	r		r	r	r	r	r		r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
												r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **ITAMP13MF**: internal tamper 13 secure interrupt masked flag
This flag is set by hardware when the internal tamper 13 secure interrupt is raised.

Bit 27 **ITAMP12MF**: internal tamper 12 secure interrupt masked flag
This flag is set by hardware when the internal tamper 12 secure interrupt is raised.

Bit 26 **ITAMP11MF**: internal tamper 11 secure interrupt masked flag
This flag is set by hardware when the internal tamper 11 secure interrupt is raised.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9MF**: internal tamper 9 secure interrupt masked flag
This flag is set by hardware when the internal tamper 9 secure interrupt is raised.

Bit 23 **ITAMP8MF**: Internal tamper 8 secure interrupt masked flag
This flag is set by hardware when the internal tamper 8 secure interrupt is raised.

Bit 22 **ITAMP7MF**: Internal tamper 7 secure interrupt masked flag
This flag is set by hardware when the internal tamper 7 secure interrupt is raised.

Bit 21 **ITAMP6MF**: Internal tamper 6 secure interrupt masked flag
This flag is set by hardware when the internal tamper 6 secure interrupt is raised.

Bit 20 **ITAMP5MF**: Internal tamper 5 secure interrupt masked flag
This flag is set by hardware when the internal tamper 5 secure interrupt is raised.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3MF**: Internal tamper 3 secure interrupt masked flag
This flag is set by hardware when the internal tamper 3 secure interrupt is raised.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **TAMP4MF**: TAMP4 secure interrupt masked flag
This flag is set by hardware when the tamper 4 secure interrupt is raised.

Bit 2 **TAMP3MF**: TAMP3 secure interrupt masked flag
This flag is set by hardware when the tamper 3 secure interrupt is raised.

Bit 1 **TAMP2MF**: TAMP2 secure interrupt masked flag
This flag is set by hardware when the tamper 2 secure interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 secure interrupt masked flag
This flag is set by hardware when the tamper 1 secure interrupt is raised.

34.6.15 TAMP status clear register (TAMP_SCR)

This register can be protected against nonsecure access. Refer to *Section 34.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 34.3.7: TAMP privilege protection modes*.

Address offset: 0x3C

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	CITAMP13F	CITAMP12F	CITAMP11F	Res.	CITAMP9F	CITAMP8F	CITAMP7F	CITAMP6F	CITAMP5F	Res.	CITAMP3F	Res.	Res.
			w	w	w		w	w	w	w	w		w		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP4F	CTAMP3F	CTAMP2F	CTAMP1F
												w	w	w	w

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **CITAMP13F**: Clear ITAMP13 detection flag
Writing 1 in this bit clears the ITAMP13F bit in the TAMP_SR register.

Bit 27 **CITAMP12F**: Clear ITAMP12 detection flag
Writing 1 in this bit clears the ITAMP12F bit in the TAMP_SR register.

Bit 26 **CITAMP11F**: Clear ITAMP11 detection flag
Writing 1 in this bit clears the ITAMP11F bit in the TAMP_SR register.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **CITAMP9F**: Clear ITAMP9 detection flag
Writing 1 in this bit clears the ITAMP9F bit in the TAMP_SR register.

Bit 23 **CITAMP8F**: Clear ITAMP8 detection flag
Writing 1 in this bit clears the ITAMP8F bit in the TAMP_SR register.

Bit 22 **CITAMP7F**: Clear ITAMP7 detection flag
Writing 1 in this bit clears the ITAMP7F bit in the TAMP_SR register.

Bit 21 **CITAMP6F**: Clear ITAMP6 detection flag
Writing 1 in this bit clears the ITAMP6F bit in the TAMP_SR register.

Bit 20 **CITAMP5F**: Clear ITAMP5 detection flag
Writing 1 in this bit clears the ITAMP5F bit in the TAMP_SR register.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **CITAMP3F**: Clear ITAMP3 detection flag
Writing 1 in this bit clears the ITAMP3F bit in the TAMP_SR register.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **CTAMP4F**: Clear TAMP4 detection flag
Writing 1 in this bit clears the TAMP4F bit in the TAMP_SR register.

Bit 2 **CTAMP3F**: Clear TAMP3 detection flag
Writing 1 in this bit clears the TAMP3F bit in the TAMP_SR register.

Bit 1 **CTAMP2F**: Clear TAMP2 detection flag
Writing 1 in this bit clears the TAMP2F bit in the TAMP_SR register.

Bit 0 **CTAMP1F**: Clear TAMP1 detection flag
Writing 1 in this bit clears the TAMP1F bit in the TAMP_SR register.

34.6.16 TAMP monotonic counter 1 register (TAMP_COUNT1R)

This register can be protected against nonsecure access. Refer to *Section 34.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 34.3.7: TAMP privilege protection modes*.

Address offset: 0x040

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 **COUNT[31:0]:** Monotonic counter

The monotonic counter increments by one when a write access to this register occurs, regardless of the written value. Reading this register returns the monotonic counter value. This register does not roll-over and freezes upon reaching its maximum value.

34.6.17 TAMP resources protection configuration register (TAMP_RPCFGR)

This register can be protected against nonsecure access. Refer to *Section 34.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 34.3.7: TAMP privilege protection modes*.

Address offset: 0x54

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RP CFG5	RP CFG4	RP CFG3	Res.	RP CFG1	Res.
										rw	rw	rw		rw	

Bit 31 Reserved, must be kept at reset value.

Bits 30:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **RPCFG5**: Configurable resource 5 protection(1)
0: Resource 5 is not included in the device secrets protected by TAMP peripheral
1: Resource 5 is included in the device secrets protected by TAMP peripheral

Bit 4 **RPCFG4**: Configurable resource 4 protection(2)
0: Resource 4 is not included in the device secrets protected by TAMP peripheral
1: Resource 4 is included in the device secrets protected by TAMP peripheral

Bit 3 **RPCFG3**: Configurable resource 3 protection(3)
0: Resource 3 is not included in the device secrets protected by TAMP peripheral
1: Resource 3 is included in the device secrets protected by TAMP peripheral

Bit 2 Reserved, must be kept at reset value.

Bit 1 **RPCFG1**: Configurable resource 1 protection(4)
0: Resource 1 is not included in the device secrets protected by TAMP peripheral
1: Resource 1 is included in the device secrets protected by TAMP peripheral

Bit 0 Reserved, must be kept at reset value.

1. Refer to tamp_confirmed_rpcfg5 and tamp_potential_rpcfg5 signals in *Table 312: TAMP input/output pins* and *Table 314: TAMP interconnection*.
2. Refer to tamp_confirmed_rpcfg4 and tamp_potential_rpcfg4 signals in *Table 312: TAMP input/output pins* and *Table 314: TAMP interconnection*.
3. Refer to tamp_confirmed_rpcfg3 and tamp_potential_rpcfg3 signals in *Table 312: TAMP input/output pins* and *Table 314: TAMP interconnection*.
4. Refer to tamp_confirmed_rpcfg1 and tamp_potential_rpcfg1 signals in *Table 312: TAMP input/output pins* and *Table 314: TAMP interconnection*.

34.6.18 TAMP backup x register (TAMP_BKPxR)

This register can be protected against nonsecure access. Refer to *Section 34.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 34.3.7: TAMP privilege protection modes*.

Address offset: 0x100 + 0x04 * x, (x = 0 to 31)

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 **BKP[31:0]:**
The application can write or read data to and from these registers.
In the default (ERASE) configuration this register is reset on a tamper detection event. It is forced to reset value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

34.6.19 TAMP register map

Table 322. TAMP register map and reset values

Offset	Register name	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
0x00	TAMP_CR1	Res.	Res.	Res.	ITAMP13E	ITAMP12E	ITAMP11E	Res.	ITAMP9E	ITAMP8E	ITAMP7E	ITAMP6E	ITAMP5E	Res.	ITAMP3E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP4E	TAMP3E	TAMP2E	TAMP1E
	Reset value				0	0	0		0	0	0	0	0		0															0	0	0	0
0x04	TAMP_CR2	Res.	Res.	Res.	Res.	TAMP4TRG	TAMP3TRG	TAMP2TRG	TAMP1TRG	BKERASE	BKBLOCK	Res.	Res.	Res.	TAMP3MSK	TAMP2MSK	TAMP1MSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP4POM	TAMP3POM	TAMP2POM	TAMP1POM
	Reset value					0	0	0	0	0	0				0	0	0													0	0	0	0
0x08	TAMP_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP13POM	ITAMP12POM	ITAMP11POM	Res.	ITAMP9POM	ITAMP8POM	ITAMP7POM	ITAMP6POM	ITAMP5POM	Res.	ITAMP3POM	Res.	Res.
	Reset value																				0	0	0		0	0	0	0	0		0		
0x0C	TAMP_FLTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]		
	Reset value																									0	0	0	0	0	0	0	0
0x10	TAMP_ATCR1	FLTEN	ATOSHARE	Res.	Res.	Res.	AT PER[2:0]			Res.	Res.	Res.	Res.	Res.	ATCK SEL[2:0]			ATO SEL4 [1:0]		ATO SEL3 [1:0]		ATO SEL2 [1:0]		ATO SEL1 [1:0]		Res.	Res.	Res.	Res.	TAMP4AM	TAMP3AM	TAMP2AM	TAMP1AM
	Reset value	0	0				0	0	0						1	1	1	0	0	0	0	0	0	0	0					0	0	0	0
0x14	TAMP_ ATSEEDR	SEED[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	TAMP_ATOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INITS	SEEDF	Res.	Res.	Res.	Res.	Res.	Res.	PRNG[7:0]							
	Reset value																	0	0							0	0	0	0	0	0	0	0
0x1C	TAMP_ATCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATO SEL4 [2:0]			ATO SEL3 [2:0]			ATO SEL2 [2:0]			ATO SEL1 [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0								
0x20	TAMP_SEC CFGR	TAMPSEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPWSEC[7:0]								CNT1SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRWSEC[7:0]							
	Reset value	0								0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0
0x24	TAMP_ PRIVCFGR	TAMPPRIV	BKPWPRIV	BKPRWPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT1PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0														0															

Table 322. TAMP register map and reset values (continued)

Offset	Register name	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
0x2C	TAMP_IER	Res.	Res.	Res.	ITAMP13IE	ITAMP12IE	ITAMP11IE	Res.	ITAMP9IE	ITAMP8IE	ITAMP7IF	ITAMP6IE	ITAMP5IE	Res.	ITAMP3IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP4IE	TAMP3IE	TAMP2IE	TAMP1IE
	Reset value				0	0	0		0	0	0	0	0		0															0	0	0	0
0x30	TAMP_SR	Res.	Res.	Res.	ITAMP13F	ITAMP12F	ITAMP11F	Res.	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6F	ITAMP5F	Res.	ITAMP3F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP4F	TAMP3F	TAMP2F	TAMP1F
	Reset value				0	0	0		0	0	0	0	0		0															0	0	0	0
0x34	TAMP_MISR	Res.	Res.	Res.	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	Res.	ITAMP3MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value				0	0	0		0	0	0	0	0		0															0	0	0	0
0x38	TAMP_SMISR	Res.	Res.	Res.	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	Res.	ITAMP3MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value				0	0	0		0	0	0	0	0		0	0														0	0	0	0
0x3C	TAMP_SCR	Res.	Res.	Res.	CITAMP13F	CITAMP12F	CITAMP11F	Res.	CITAMP9F	CITAMP8F	CITAMP7F	CITAMP6F	CITAMP5F	Res.	CITAMP3F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP4F	CTAMP3F	CTAMP2F	CTAMP1F
	Reset value				0	0	0		0	0	0	0	0		0															0	0	0	0
0x40	TAMP_ COUNT1R	COUNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54	TAMP_ RPCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPCFG5	RPCFG4	RPCFG3	Res.	RPCFG1	Res.
	Reset value																											0	0	0		0	
0x100 + 0x04*x, (x= 0 to 31)	TAMP_BKPxR	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

35 Inter-integrated circuit interface (I2C)

35.1 I2C introduction

The I2C peripheral handles the interface between the device and the serial I²C (inter-integrated circuit) bus. It provides multicontroller capability, and controls all I²C-bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

The I2C peripheral is also SMBus (system management bus) and PMBus® (power management bus) compatible.

It can use DMA to reduce the CPU load.

35.2 I2C main features

- I²C-bus specification rev03 compatibility:
  - Target and controller modes
  - Multicontroller capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit target addresses (2 addresses, 1 with configurable mask)
  - All 7-bit-addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy-to-use event management
  - Clock stretching (optional)
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters
- SMBus specification rev 3.0 compatibility[(a)]:
  - Hardware PEC (packet error checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.3 standard compatibility
- Independent clock

a. To check the compliance of the GPIOs selected for SMBus with the specified logical levels, refer to the product datasheet.

- Wake-up from Stop mode
- Autonomous functionality in Stop mode

For information on I2C instantiation, refer to *Section 35.3: I2C implementation*.

35.3 I2C implementation

This section provides an implementation overview with respect to the I2C instantiation.

Table 323. I2C implementation

I2C features(1)	I2C1	I2C3
7-bit addressing mode	X	X
10-bit addressing mode	X	X
Standard-mode (up to 100 kbit/s)	X	X
Fast-mode (up to 400 kbit/s)	X	X
Fast-mode Plus with 20 mA output drive I/Os (up to 1 Mbit/s)	X	X
Independent clock	X	X
Autonomous mode	X	X
Wake-up from Stop mode only (no autonomous mode)	X	X
SMBus/PMBus	X	X

1. X = supported.

35.4 I2C functional description

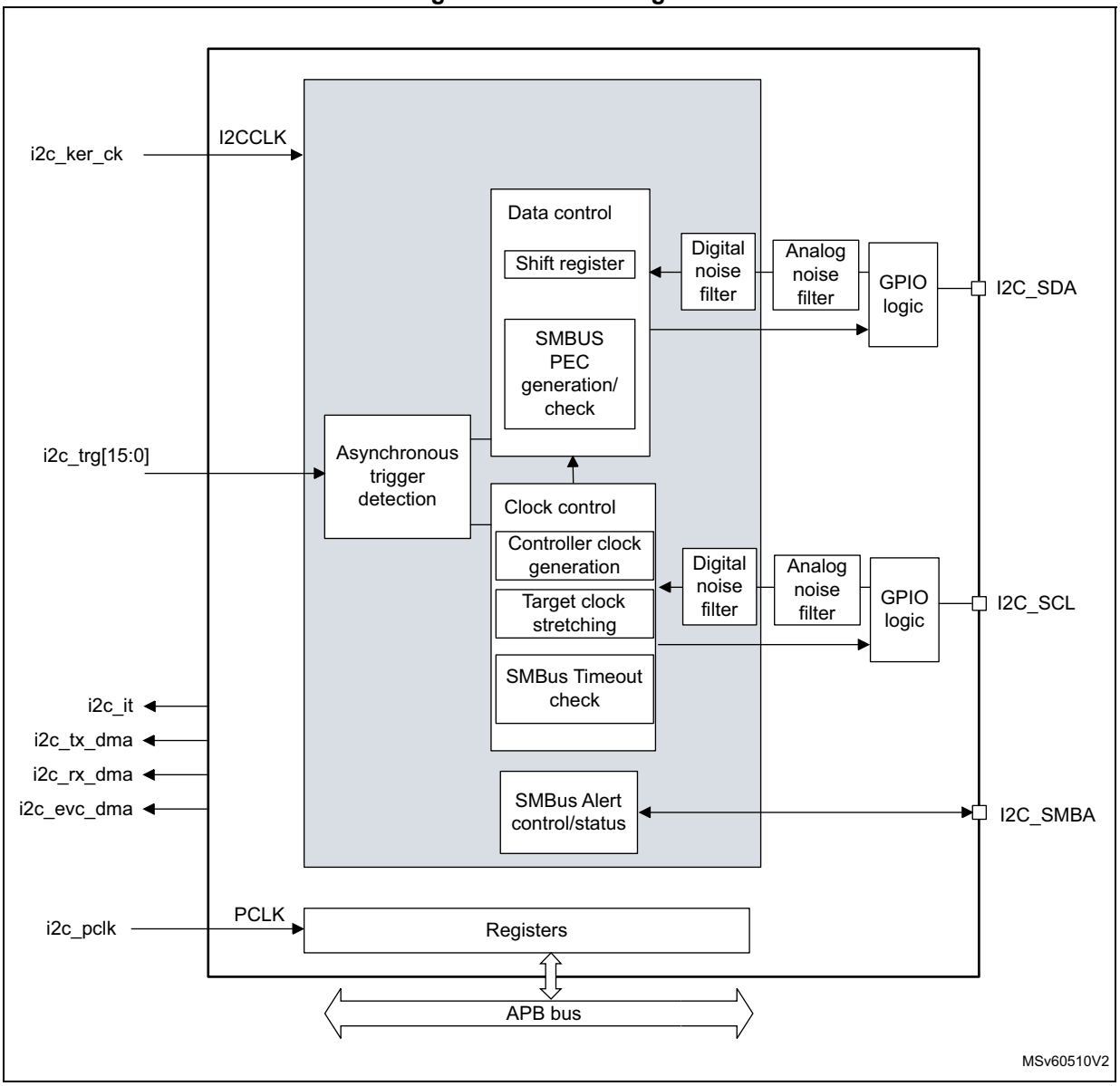
In addition to receiving and transmitting data, the peripheral converts them from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The peripheral is connected to the I²C-bus through a data pin (SDA) and a clock pin (SCL). It supports Standard-mode (up to 100 kHz), Fast-mode (up to 400 kHz), and Fast-mode Plus (up to 1 MHz) I²C-bus.

The peripheral can also be connected to an SMBus, through the data pin (SDA), the clock pin (SCL), and an optional SMBus alert pin (SMBA).

The independent clock function allows the I2C communication speed to be independent of the i2c_pclk frequency.

35.4.1 I2C block diagram

Figure 276. Block diagram



35.4.2 I2C pins and internal signals

Table 324. I2C input/output pins

Pin name	Signal type	Description
I2C_SDA	Bidirectional	I²C-bus data
I2C_SCL	Bidirectional	I²C-bus clock
I2C_SMBA	Bidirectional	SMBus alert

Table 325. I2C internal input/output signals

Internal signal name	Signal type	Description
i2c_ker_ck	Input	I2C kernel clock, also named I2CCLK in this document
i2c_pclk	Input	I2C APB clock
i2c_trg[15:0]	Input	I2C triggers
i2c_it	Output	I2C interrupts, refer to *Table 340* for the list of interrupt sources
i2c_rx_dma	Output	I2C receive data DMA request (I2C_RX)
i2c_tx_dma	Output	I2C transmit data DMA request (I2C_TX)
i2c_evc_dma	Output	I2C event control DMA request (I2C_EVC)

Table 326. I2C1 interconnection

Signal name	Source/destination
i2c_trg0	lpdma1_ch0_tc
i2c_trg1	lpdma1_ch1_tc
i2c_trg2	lpdma1_ch2_tc
i2c_trg3	lpdma1_ch3_tc
i2c_trg4	exti5
i2c_trg5	exti9
i2c_trg6	lptim1_ch1
i2c_trg7	lptim2_ch1
i2c_trg8	Reserved
i2c_trg9	Reserved
i2c_trg10	rtc_alra_trg
i2c_trg11	rtc_wut_trg
i2c_trg12..15	Reserved

Table 327. I2C3 interconnection

Signal name	Source/destination
i2c_trg0	lpdma1_ch0_tc
i2c_trg1	lpdma1_ch1_tc
i2c_trg2	lpdma1_ch2_tc
i2c_trg3	lpdma1_ch3_tc
i2c_trg4	exti5
i2c_trg5	exti8
i2c_trg6	lptim1_ch1
i2c_trg7	Reserved

Table 327. I2C3 interconnection (continued)

Signal name	Source/destination
i2c_trg8	Reserved
i2c_trg9	Reserved
i2c_trg10	rtc_alra_trg
i2c_trg11	rtc_wut_trg
i2c_trg12..15	Reserved

35.4.3 I2C clock requirements

The I2C kernel is clocked by i2c_ker_ck.

The i2c_ker_ck period t_{I2CCLK} must respect the following conditions:

$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4$

$t_{I2CCLK} < t_{HIGH}$

where t_{LOW} is the SCL low time, t_{HIGH} is the SCL high time, and $t_{filters}$ is the sum of the analog and digital filter delays (when enabled).

The digital filter delay is DNF[3:0] x t_{I2CCLK}.

The i2c_pclk clock period t_{PCLK} must respect the condition $t_{PCLK} < 4/3\ t_{SCL}$, where t_{SCL} is the SCL period.

Caution: When the I2C kernel is clocked by i2c_pclk, this clock must respect the conditions for t_{I2CCLK}.

35.4.4 I2C mode selection

The peripheral can operate as:

- Target transmitter
- Target receiver
- Controller transmitter
- Controller receiver

By default, the peripheral operates in target mode. It automatically switches from target to controller mode upon generating START condition, and from controller to target mode upon arbitration loss or upon generating STOP condition. This allows the use of the I2C peripheral in a multicontroller I²C-bus environment.

Communication flow

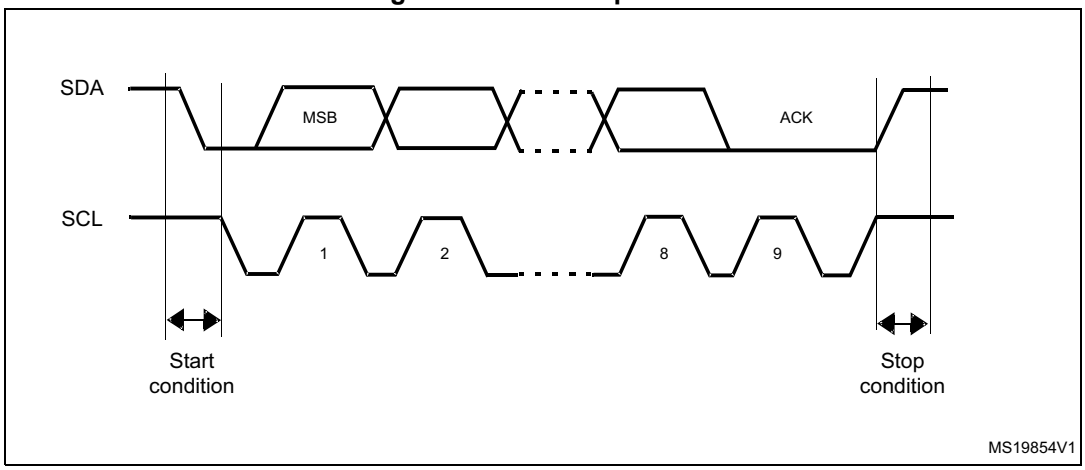
In controller mode, the I2C peripheral initiates a data transfer and generates the clock signal. Serial data transfers always begin with a START condition and end with a STOP condition. Both START and STOP conditions are generated in controller mode by software.

In target mode, the peripheral recognizes its own 7-bit or 10-bit address, and the general call address. The general call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The address is contained in the first byte (7-bit addressing) or in the first two bytes (10-bit addressing) following the START condition. The address is always transmitted in controller mode.

The following figure shows the transmission of a single byte. The controller generates nine SCL pulses. The transmitter sends the eight data bits to the receiver with the SCL pulses 1 to 8. Then the receiver sends the acknowledge bit to the transmitter with the ninth SCL pulse.

Figure 277. I²C-bus protocol



The acknowledge can be enabled or disabled by software. The own addresses of the I2C peripheral can be selected by software.

35.4.5 I2C initialization

Enabling and disabling the peripheral

Before enabling the I2C peripheral, configure and enable its clock through the RCC, and initialize its control registers.

The I2C peripheral can then be enabled by setting the PE bit of the I2C_CR1 register.

Disabling the I2C peripheral by clearing the PE bit resets the I2C peripheral. Refer to *Section 35.4.6* for more details.

Noise filters

Before enabling the I2C peripheral by setting the PE bit of the I2C_CR1 register, the user must configure the analog and/or digital noise filters, as required.

The analog noise filter on the SDA and SCL inputs complies with the I²C-bus specification which requires, in Fast-mode and Fast-mode Plus, the suppression of spikes shorter than 50 ns. Enabled by default, it can be disabled by setting the ANFOFF bit.

The digital filter is controlled through the DNF[3:0] bitfield of the I2C_CR1 register. When it is enabled, the internal SCL and SDA signals only take the level of their corresponding I²C-bus line when remaining stable for more than DNF[3:0] periods of i2c_ker_ck. This allows suppressing spikes shorter than the filtering capacity period programmable from one to fifteen i2c_ker_ck periods.

The following table compares the two filters.

Table 328. Comparison of analog and digital filters

Item	Analog filter	Digital filter
Filtering capacity(1)	≥ 50 ns	One to fifteen i2c_ker_ck periods
Benefits	Available in Stop mode	– Programmable filtering capacity – Extra filtering capability versus I²C-bus specification requirements – Stable filtering capacity
Drawbacks	Filtering capacity variation with temperature, voltage, and silicon process	Functionality in Stop mode not supported when the digital filter is enabled

1. Maximum duration of spikes that the filter can suppress

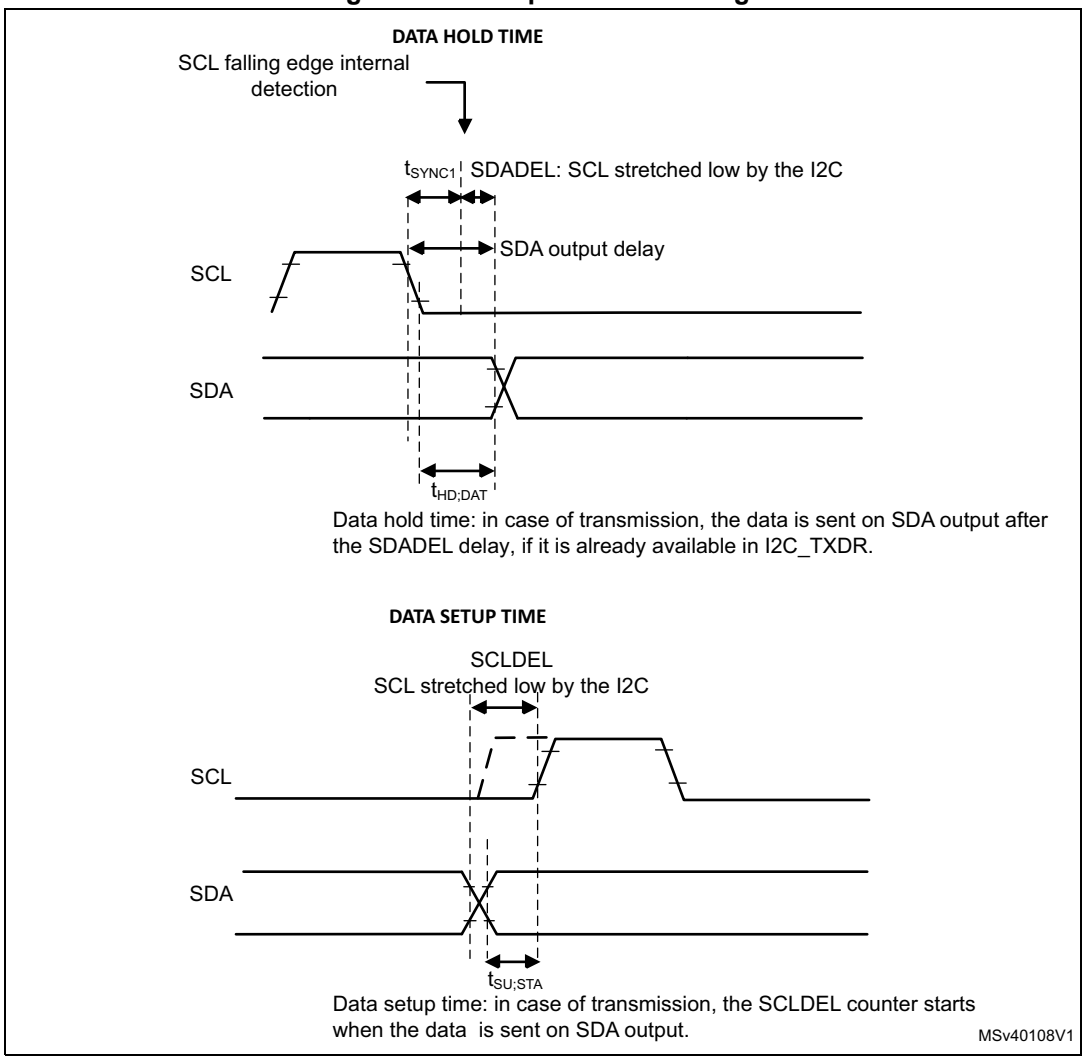
Caution: The filter configuration cannot be changed when the I2C peripheral is enabled.

I2C timings

To ensure correct data hold and setup times, the corresponding timings must be configured through the PRESC[3:0], SCLDEL[3:0], and SDADEL[3:0] bitfields of the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C configuration* window.

Figure 278. Setup and hold timings



When the SCL falling edge is internally detected, the delay t_{SDADEL} (impacting the hold time $t_{HD;DAT}$) is inserted before sending SDA output:

t_{SDADEL} = SDADEL x t_{PRESC} + t_{I2CCLK}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}.

The total SDA output delay is:

t_{SYNC1} + {[SDADEL x (PRESC + 1) + 1] x t_{I2CCLK}}

The t_{SYNC1} duration depends upon:

- SCL falling slope
- input delay $t_{AF(min)} < t_{AF} < t_{AF(max)}$ introduced by the analog filter (if enabled)
- input delay t_{DNF} = DNF x t_{I2CCLK} introduced by the digital filter (if enabled)
- delay due to SCL synchronization to i2c_ker_ck clock (two to three i2c_ker_ck periods)

To bridge the undefined region of the SCL falling edge, the user must set SDADEL[3:0] so as to fulfill the following condition:

{$t_{f(max)}$ + $t_{HD;DAT(min)}$ - $t_{AF(min)}$ - [(DNF + 3) x t_{I2CCLK}]} / {(PRESC + 1) x t_{I2CCLK}} ≤ SDADEL

SDADEL ≤ {$t_{HD;DAT\ (max)}$ - $t_{AF(max)}$ - [(DNF + 4) x t_{I2CCLK}]} / {(PRESC + 1) x t_{I2CCLK}}

Note: $t_{AF(min)}$ and $t_{AF(max)}$ are only part of the condition when the analog filter is enabled. Refer to the device datasheet for t_{AF} values.

The $t_{HD;DAT}$ time can at maximum be 3.45 µs for Standard-mode, 0.9 µs for Fast-mode, and 0.45 µs for Fast-mode Plus. It must be lower than the maximum of $t_{VD;DAT}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. When it stretches SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case. The previous condition then becomes:

$SDADEL \leq \{t_{VD;DAT\ (max)} - t_{r\ (max)} - t_{AF\ (max)} - [(DNF + 4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$

Note: This condition can be violated when NOSTRETCH = 0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL[3:0] value.

After t_{SDADEL}, or after sending SDA output when the target had to stretch the clock because the data was not yet written in I2C_TXDR register, the SCL line is kept at low level during the setup time. This setup time is $t_{SCLDEL} = (SCLDEL + 1) \times t_{PRESC}$, where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$. t_{SCLDEL} impacts the setup time $t_{SU;DAT}$.

To bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL[3:0] so as to fulfill the following condition:

$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC + 1) \times t_{I2CCLK}]\} - 1 \leq SCLDEL$

Refer to the following table for t_f, t_r, $t_{HD;DAT}$, $t_{VD;DAT}$, and $t_{SU;DAT}$ standard values.

Use the SDA and SCL real transition time values measured in the application to widen the scope of allowed SDADEL[3:0] and SCLDEL[3:0] values. Use the maximum SDA and SCL transition time values defined in the standard to make the device work reliably regardless of the application.

Note: At every clock pulse, after SCL falling edge detection, I2C operating as controller or target stretches SCL low during at least [(SDADEL + SCLDEL + 1) x (PRESC + 1) + 1] x t_{I2CCLK}, in both transmission and reception modes. In transmission mode, if the data is not yet written in I2C_TXDR when SDA delay elapses, the I2C peripheral keeps stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

When the NOSTRETCH bit is set in target mode, the SCL is not stretched. The SDADEL[3:0] must then be programmed so that it ensures a sufficient setup time.

Table 329. I²C-bus and SMBus specification data setup and hold times

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	µs
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100	-	50	-	250	-	ns
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Additionally, in controller mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0], and SCLL[7:0] bitfields of the I2C_TIMINGR register.

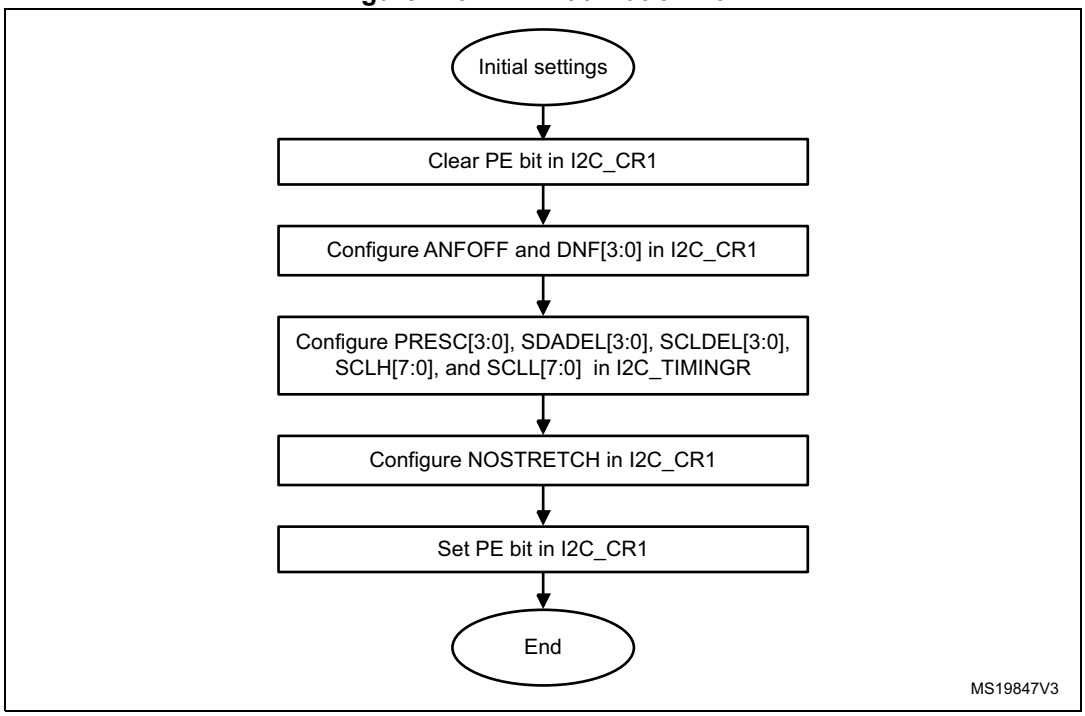
When the SCL falling edge is internally detected, the I2C peripheral releasing the SCL output after the delay t_{SCLL} = (SCLL + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLL} delay impacts the SCL low time t_{LOW}.

When the SCL rising edge is internally detected, the I2C peripheral forces the SCL output to low level after the delay t_{SCLH} = (SCLH + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLH} impacts the SCL high time t_{HIGH}.

Refer to *I2C controller initialization* for more details.

Caution: Changing the timing configuration and the NOSTRETCH configuration is not allowed when the I2C peripheral is enabled. Like the timing settings, the target NOSTRETCH settings must also be done before enabling the peripheral. Refer to *I2C target initialization* for more details.

Figure 279. I2C initialization flow



35.4.6 I2C reset

The reset of the I2C peripheral is performed by clearing the PE bit of the I2C_CR1 register. It has the effect of releasing the SCL and SDA lines. Internal state machines are reset and the communication control bits and the status bits revert to their reset values. This reset does not impact the configuration registers.

The impacted register bits are:

1. I2C_CR2 register: START, STOP, PECBYTE, and NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, PECERR, TIMEOUT, ALERT, and OVR

PE must be kept low during at least three APB clock cycles to perform the I2C reset. To ensure this, perform the following software sequence:

1. Write PE = 0
2. Check PE = 0
3. Write PE = 1

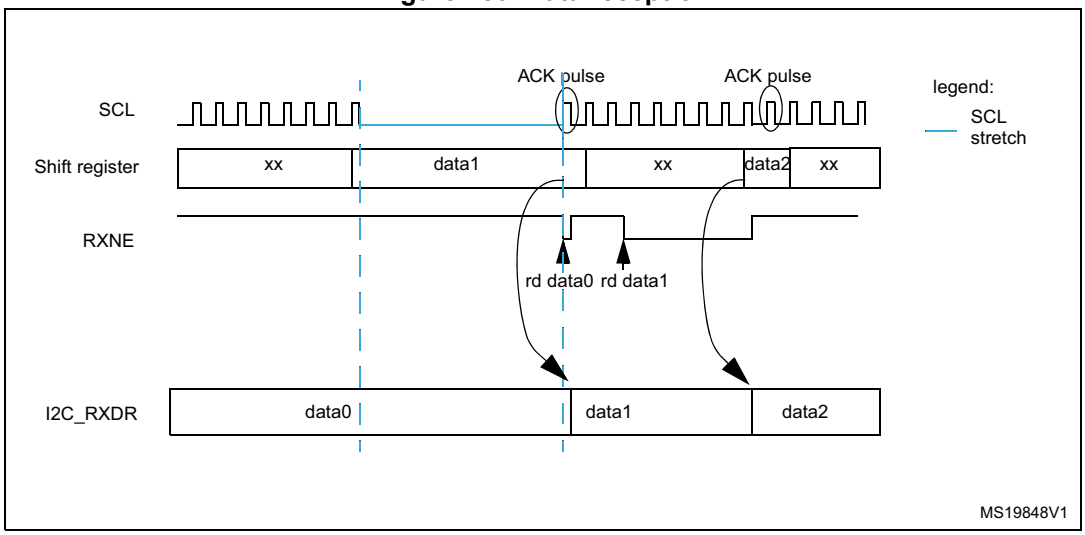
35.4.7 I2C data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the eighth SCL pulse (when the complete data byte is received), the shift register is copied into the I2C_RXDR register if it is empty (RXNE = 0). If RXNE = 1, which means that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch occurs between the eighth and the ninth SCL pulse (before the acknowledge pulse).

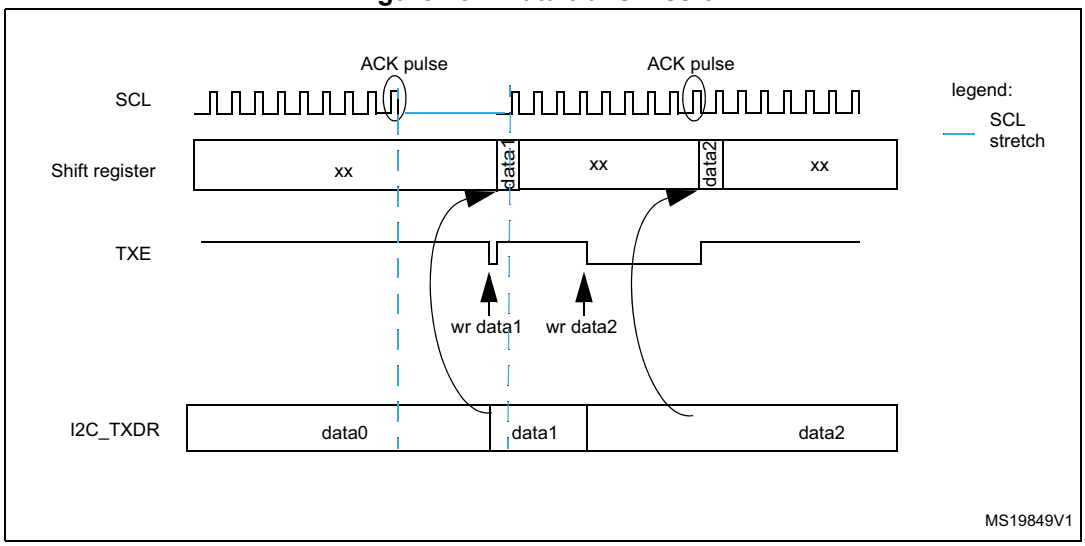
Figure 280. Data reception



Transmission

If the I2C_TXDR register is not empty (TXE = 0), its content is copied into the shift register after the ninth SCL pulse (the acknowledge pulse). Then the shift register content is shifted out on the SDA line. If TXE = 1, which means that no data is written yet in I2C_TXDR, the SCL line is stretched low until I2C_TXDR is written. The stretch starts after the ninth SCL pulse.

Figure 281. Data transmission



Hardware transfer management

The I2C features an embedded byte counter to manage byte transfer and to close the communication in various modes, such as:

- NACK, STOP and ReSTART generation in controller mode
- ACK control in target receiver mode
- PEC generation/checking

In controller mode, the byte counter is always used. By default, it is disabled in target mode. It can be enabled by software, by setting the SBC (target byte control) bit of the I2C_CR1 register.

The number of bytes to transfer is programmed in the NBYTES[7:0] bitfield of the I2C_CR2 register. If this number is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected, by setting the RELOAD bit of the I2C_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTES[7:0] is transferred (when the associated counter reaches zero), and an interrupt is generated if TCIE is set. SCL is stretched as long as the TCR flag is set. TCR is cleared by software when NBYTES[7:0] is written to a non-zero value.

When NBYTES[7:0] is reloaded with the last number of bytes to transfer, the RELOAD bit must be cleared.

When RELOAD = 0 in controller mode, the counter can be used in two modes:

- **Automatic end** (AUTOEND = 1 in the I2C_CR2 register). In this mode, the controller automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred.
- **Software end** (AUTOEND = 0 in the I2C_CR2 register). In this mode, a software action is expected once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit of the I2C_CR2 register is set. This mode must be used when the controller wants to send a RESTART condition.

Caution: The AUTOEND bit has no effect when the RELOAD bit is set.

Table 330. I2C configuration

Function	SBC bit	RELOAD bit	AUTOEND bit
Controller Tx/Rx NBYTES + STOP	X	0	1
Controller Tx/Rx + NBYTES + RESTART	X	0	0
Target Tx/Rx, all received bytes ACKed	0	X	X
Target Rx with ACK control	1	1	X

35.4.8 I2C target mode

I2C target initialization

To work in target mode, the user must enable at least one target address. The I2C_OAR1 and I2C_OAR2 registers are available to program the target own addresses OA1 and OA2, respectively.

OA1 can be configured either in 7-bit (default) or in 10-bit addressing mode, by setting the OA1MODE bit of the I2C_OAR1 register.

OA1 is enabled by setting the OA1EN bit of the I2C_OAR1 register.

If an additional target addresses are required, the second target address OA2 can be configured. Up to seven OA2 LSBs can be masked, by configuring the OA2MSK[2:0] bitfield of the I2C_OAR2 register. Therefore, for OA2MSK[2:0] configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6], or OA2[7] are compared with the received address. When OA2MSK[2:0] is other than 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX) and they are not acknowledged. If OA2MSK[2:0] = 7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

When enabled through the specific bit, the reserved addresses can be acknowledged if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK[2:0] = 0.

OA2 is enabled by setting the OA2EN bit of the I2C_OAR2 register.

The general call address is enabled by setting the GCEN bit of the I2C_CR1 register.

When the I2C peripheral is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the target uses its clock stretching capability, which means that it stretches the SCL signal at low level when required, to perform software actions. If the controller does not

support clock stretching, I2C must be configured with NOSTRETCH = 1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled, the user must read the ADDCODE[6:0] bitfield of the I2C_ISR register to check which address matched. The DIR flag must also be checked to know the transfer direction.

Target with clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is zero (default), the I2C peripheral operating as an I²C-bus target stretches the SCL signal in the following situations:

- The ADDR flag is set and the received address matches with one of the enabled target addresses.
  The stretch is released when the software clears the ADDR flag by setting the ADDRCF bit.
- In transmission, the previous data transmission is completed and no new data is written in I2C_TXDR register, or the first data byte is not written when the ADDR flag is cleared (TXE = 1).
  The stretch is released when the data is written to the I2C_TXDR register.
- In reception, the I2C_RXDR register is not read yet and a new data reception is completed.
  The stretch is released when I2C_RXDR is read.
- In target byte control mode (SBC bit set) with reload (RELOAD bit set), the last data byte transfer is finished (TCR bit set).
  The stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] bitfield.
- After SCL falling edge detection.
  The stretch is released after [(SDADEL + SCLDEL + 1) x (PRESC+ 1) + 1] x t_{I2CCLK} period.

Target without clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is set, the I2C peripheral operating as an I²C-bus target does not stretch the SCL signal.

The SCL clock is not stretched while the ADDR flag is set.

In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, it ensures that the OVR status is provided, even for the first data to be transmitted.

In reception, the data must be read from the I2C_RXDR register before the ninth SCL pulse (ACK pulse) of the next data byte occurs. If not, an overrun occurs, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Target byte control mode

To allow byte ACK control in target reception mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. This is required to comply with SMBus standards.

The reload mode must be selected to allow byte ACK control in target reception mode (RELOAD = 1). To get control of each byte, NBYTES[7:0] must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the eighth and the ninth SCL pulse. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit of the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and the next byte can be received.

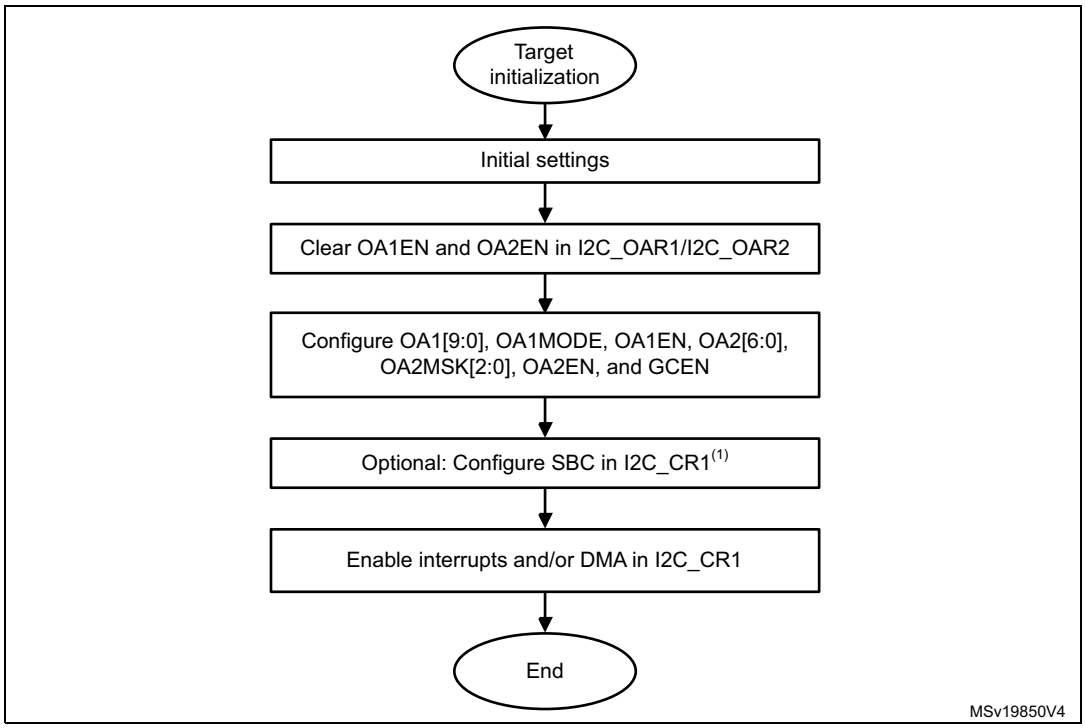
NBYTES[7:0] can be loaded with a value greater than 0x1. Receiving then continues until the corresponding number of bytes are received.

Note: The SBC bit must be configured when the I2C peripheral is disabled, when the target is not addressed, or when ADDR = 1.

The RELOAD bit value can be changed when ADDR = 1, or when TCR = 1.

Caution: The target byte control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH = 1 is not allowed.

Figure 282. Target initialization flow



1. SBC must be set to support SMBus features.

Target transmitter

A transmit interrupt status (TXIS) flag is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit of the I2C_CR1 register is set.

The TXIS flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

When NACK is received, the NACKF flag is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit of the I2C_CR1 register is set. The target automatically releases the SCL and SDA lines to let the controller perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When STOP is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag of the I2C_ISR register is set and an interrupt is generated. In most applications, the SBC bit is usually programmed to 0. In this case, if TXE = 0 when the target address is received (ADDR = 1), the user can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register, by setting the TXE bit in order to program a new data byte.

In target byte control mode (SBC = 1), the number of bytes to transmit must be programmed in NBYTES[7:0] in the address match interrupt subroutine (ADDR = 1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0].

Caution: When NOSTRETCH = 1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C_TXDR register content in the ADDR subroutine to program the first data byte. The first data byte to send must be previously programmed in the I2C_TXDR register:

- This data can be the one written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to send, the I2C_TXDR register can be flushed, by setting the TXE bit, to program a new data byte. The STOPF bit must be cleared only after these actions. This guarantees that they are executed before the first data transmission starts, following the address acknowledge.

  If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

  If a TXIS event (transmit interrupt or transmit DMA request) is required, the user must set the TXIS bit in addition to the TXE bit, to generate the event.

Figure 283. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0

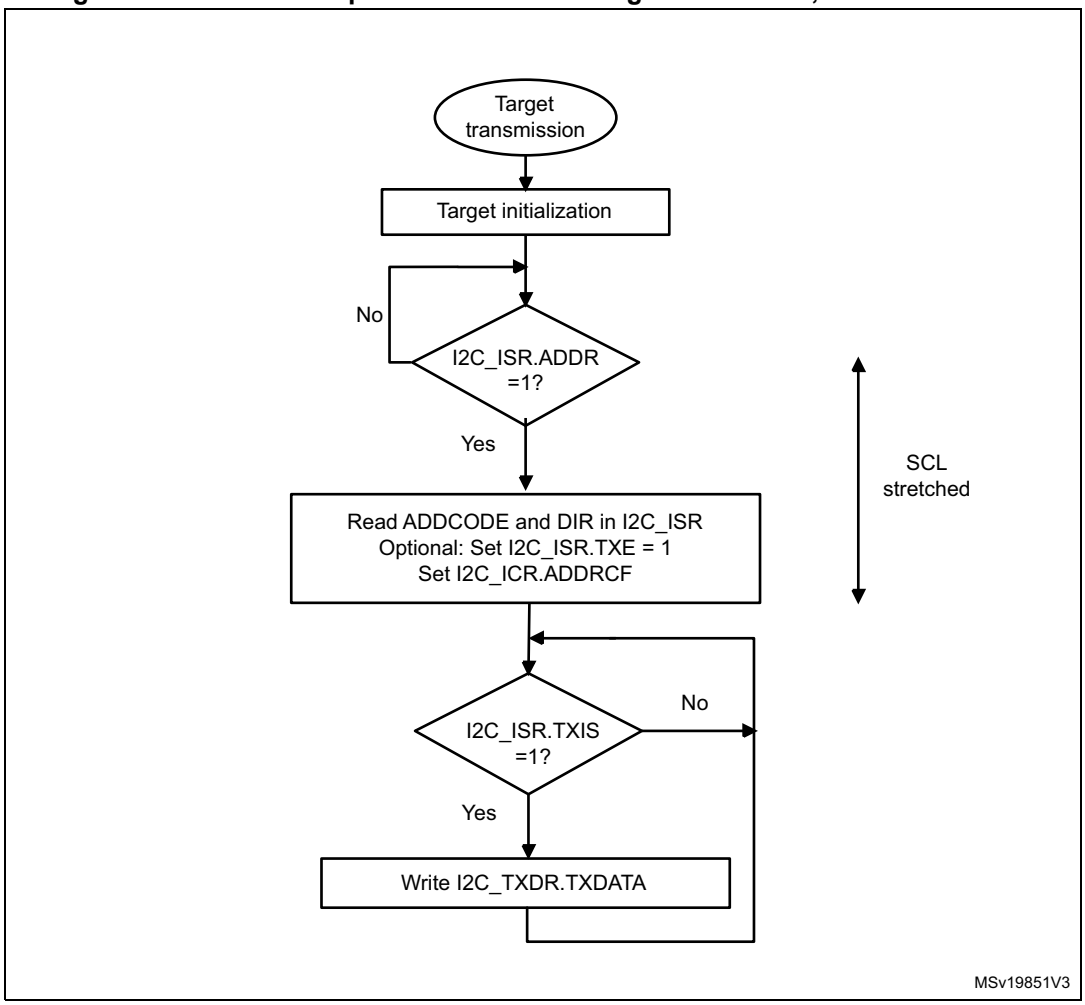


Figure 284. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1

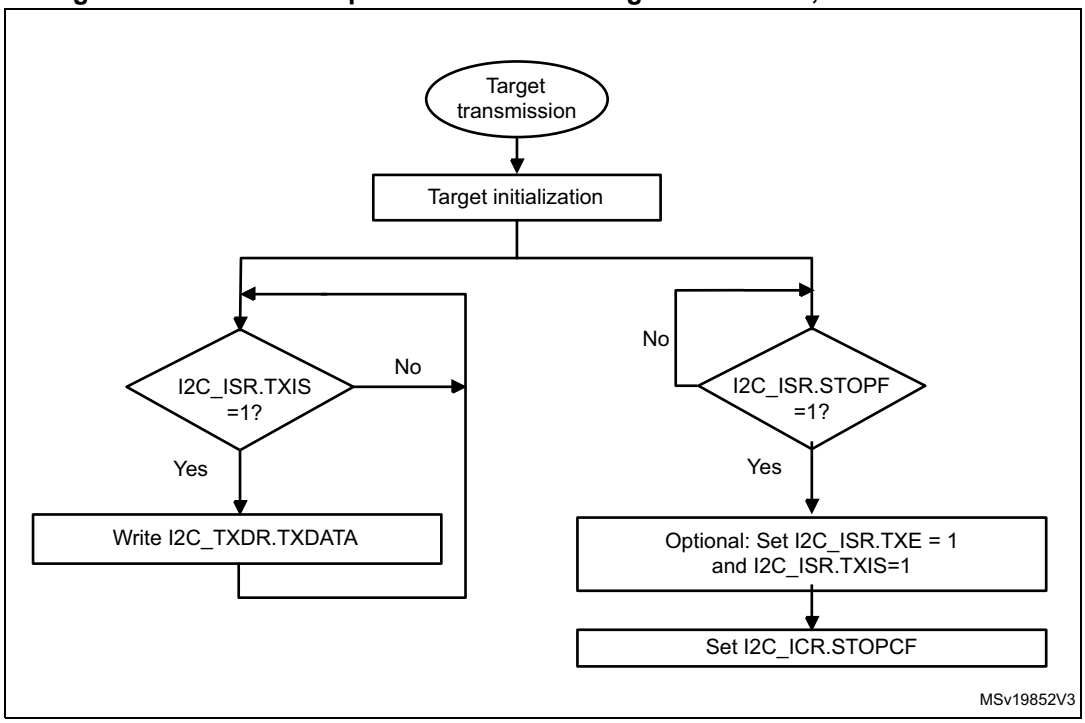
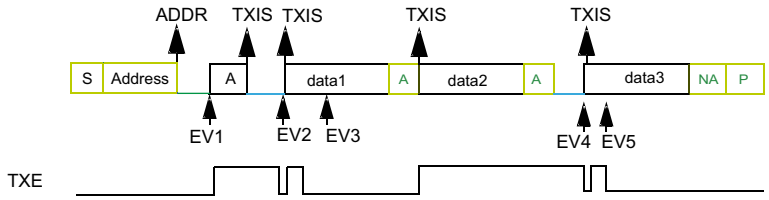


Figure 285. Transfer bus diagrams for I2C target transmitter (mandatory events only)

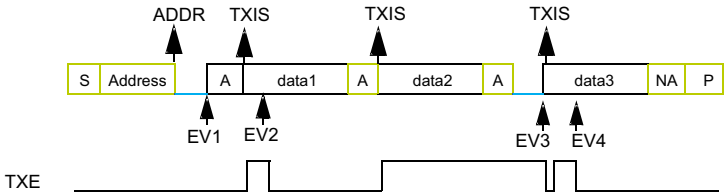
Example I2C target transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:



legend:
transmission
reception
SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2
EV4: TXIS ISR: wr data3
EV5: TXIS ISR: wr data4 (not sent)

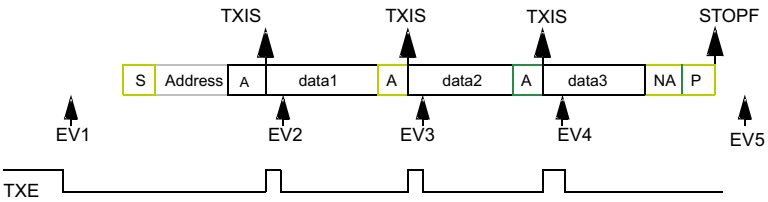
Example I2C target transmitter 3 bytes without 1st data flush, NOSTRETCH=0:



legend :
transmission
reception
SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)

Example I2C target transmitter 3 bytes, NOSTRETCH=1:



legend:
transmission
reception
SCL stretch

EV1: wr data1
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)
EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCF

MSv19853V3

Target receiver

The RXNE bit of the I2C_ISR register is set when the I2C_RXDR is full, which generates an interrupt if the RXIE bit of the I2C_CR1 register is set. RXNE is cleared when I2C_RXDR is read.

When STOP condition is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag in the I2C_ISR register is set and an interrupt is generated.

Figure 286. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0

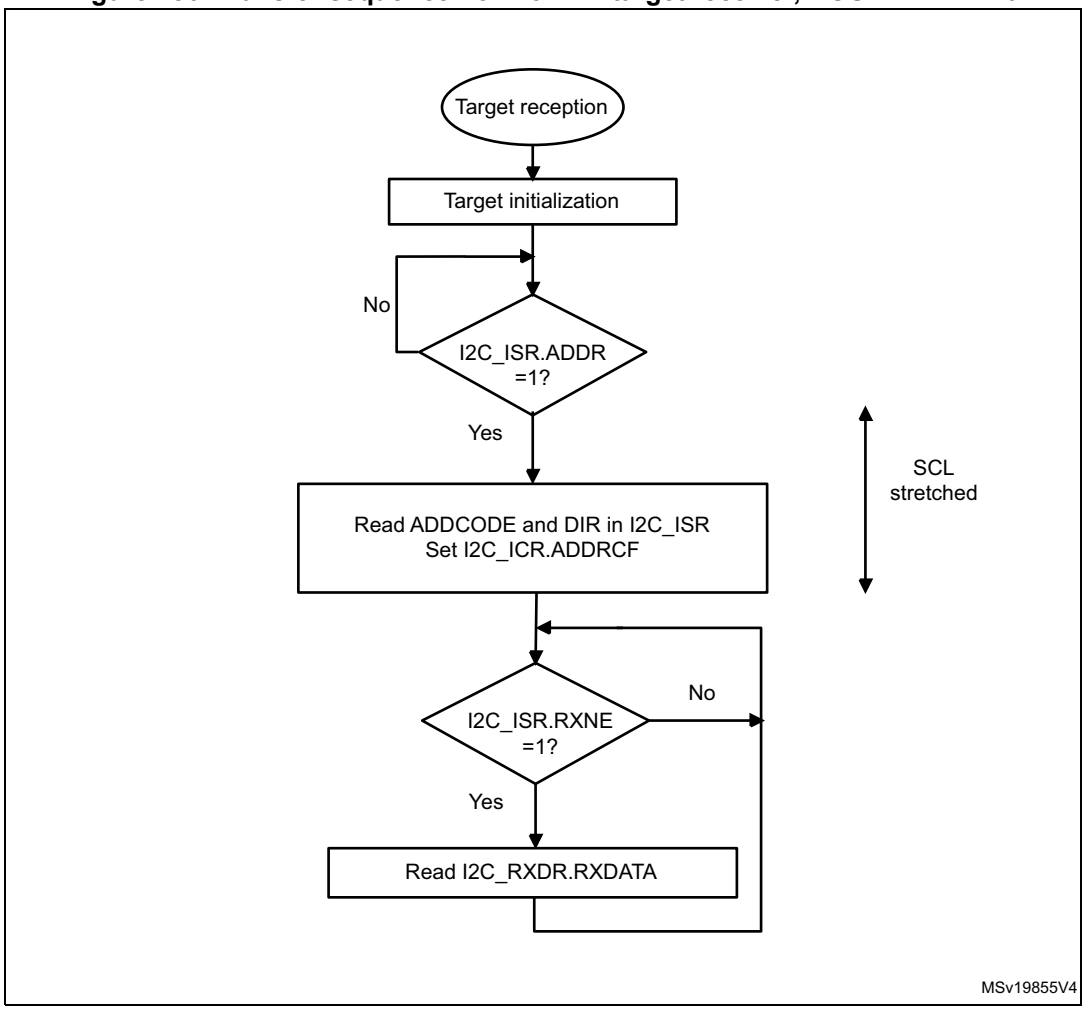


Figure 287. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1

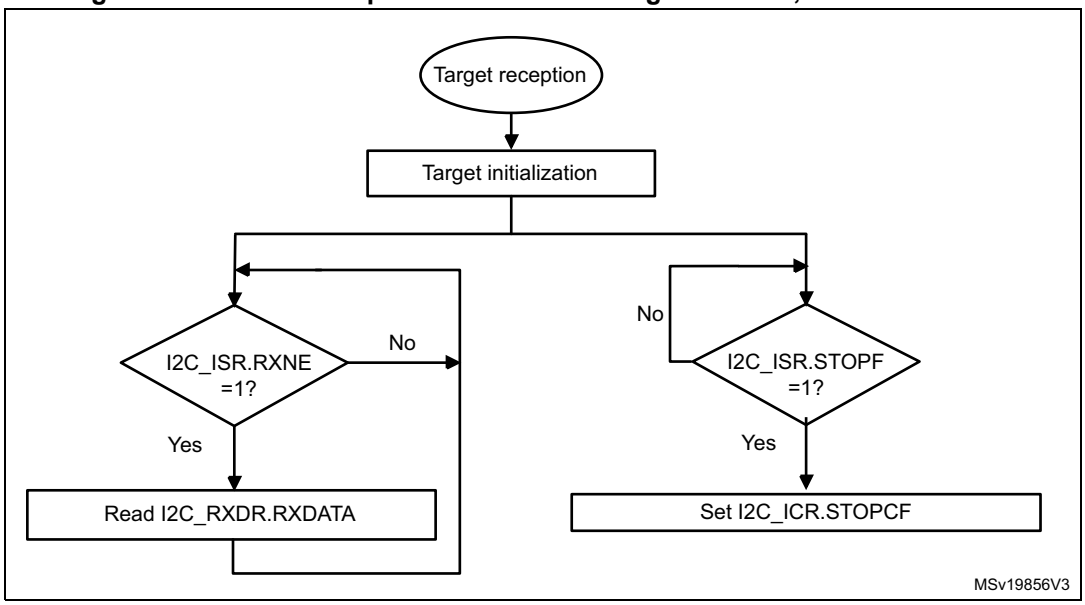
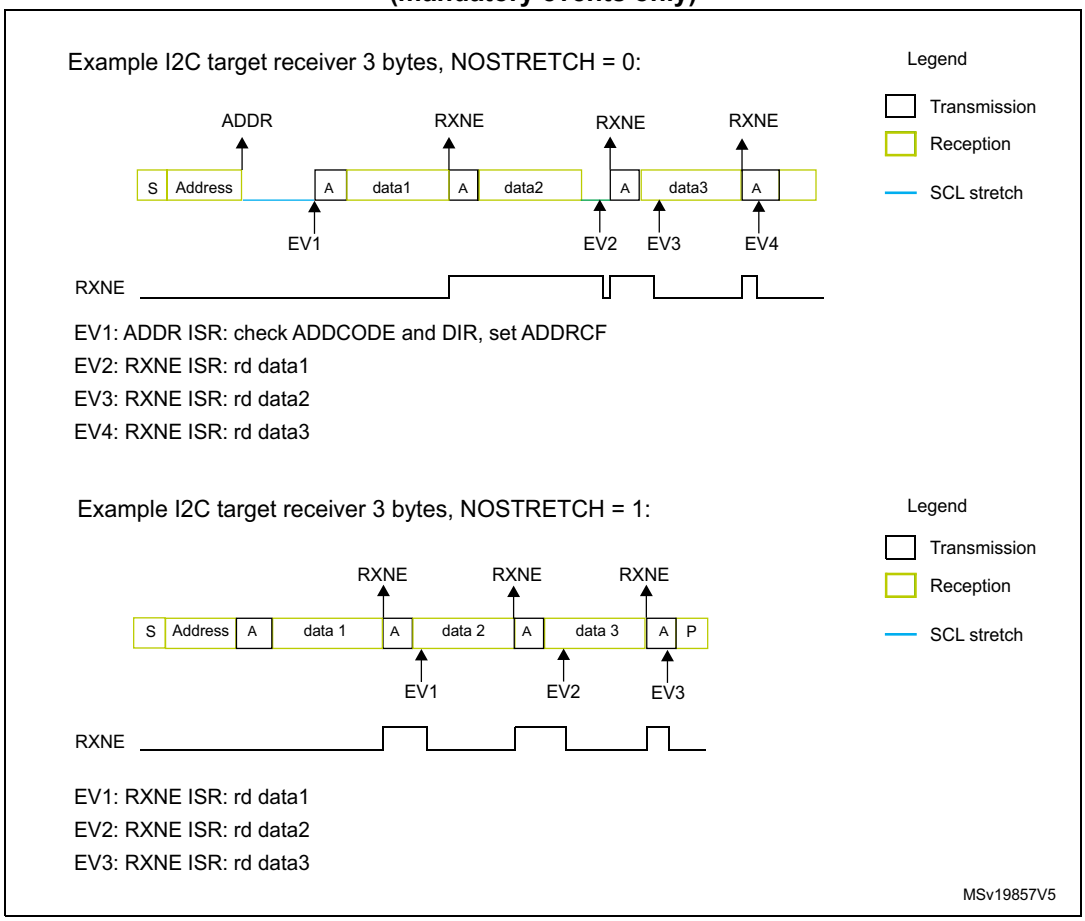


Figure 288. Transfer bus diagrams for I2C target receiver (mandatory events only)



35.4.9 I2C controller mode

I2C controller initialization

Before enabling the peripheral, the I2C controller clock must be configured, by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C Configuration* window.

A clock synchronization mechanism is implemented in order to support multicontroller environment and target clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

I2C detects its own SCL low level after a t_{SYNC1} delay depending on the SCL falling edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bitfield of the I2C_TIMINGR register.

I2C detects its own SCL high level after a t_{SYNC2} delay depending on the SCL rising edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C ties SCL to low level once the SCLH counter reaches the value programmed in the SCLH[7:0] bitfield of the I2C_TIMINGR register.

Consequently the controller clock period is:

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+ 1) + (SCLL+ 1)] \times (PRESC+ 1) \times t_{I2CCLK}\}$

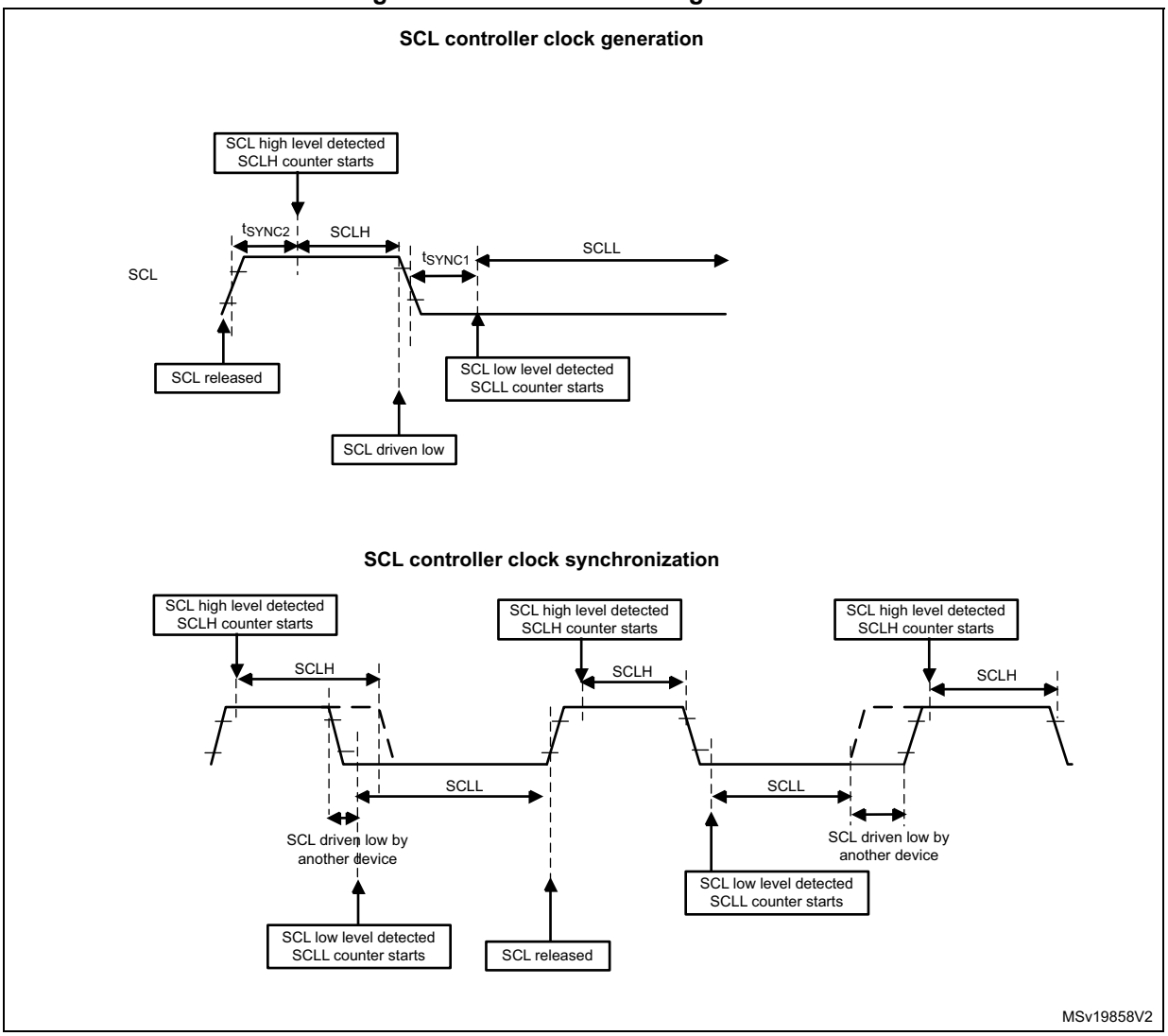
The duration of t_{SYNC1} depends upon:

- SCL falling slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

The duration of t_{SYNC2} depends upon:

- SCL rising slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

Figure 289. Controller clock generation



Caution: For compliance with the I²C-bus or SMBus specification, the controller clock must respect the timings in the following table.

Table 331. I²C-bus and SMBus specification clock timings

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency	-	100	-	400	-	1000	-	100	kHz
$t_{HD:STA}$	Hold time (repeated) START condition	4.0	-	0.6	-	0.26	-	4.0	-	µs
$t_{SU:STA}$	Set-up time for a repeated START condition	4.7	-	0.6	-	0.26	-	4.7	-	
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6	-	0.26	-	4.0	-	
t_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	
t_{LOW}	Low period of the SCL clock	4.7	-	1.3	-	0.5	-	4.7	-	
t_{HIGH}	High period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	ns
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Note: The SCLL[7:0] bitfield also determines the t_{BUF} and $t_{SU:STA}$ timings and SCLH[7:0] the $t_{HD:STA}$ and $t_{SU:STO}$ timings.

Refer to *Section 35.4.10* for examples of I2C_TIMINGR settings versus the i2c_ker_ck frequency.

Controller communication initialization (address phase)

To initiate the communication with a target to address, set the following bitfields of the I2C_CR2 register:

- ADD10: addressing mode (7-bit or 10-bit)
- SADD[9:0]: target address to send
- RD_WRN: transfer direction
- HEAD10R: in case of 10-bit address read, this bit determines whether the header only (for direction change) or the complete address sequence is sent.
- NBYTES[7:0]: the number of bytes to transfer; if equal to or greater than 255 bytes, the bitfield must initially be set to 0xFF.

Note: Changing these bitfields is not allowed as long as the START bit is set.

Before launching the communication, make sure that the I²C-bus is idle. This can be checked using the bus idle detection function or by verifying that the IDR bits of the GPIOs selected as SDA and SCL are set. Any low-level incident on the I²C-bus lines that coincides with the START condition asserted by the I2C peripheral may cause its deadlock if not filtered out by the input filters. If such incidents cannot be prevented, design the software so that it restores the normal operation of the I2C peripheral in case of a deadlock, by toggling the PE bit of the I2C_CR1 register.

To launch the communication, set the START bit of the I2C_CR2 register. The controller then automatically sends a START condition followed by the target address, either immediately if the BUSY flag is low, or t_{BUF} time after the BUSY flag transits from high to low state. The BUSY flag is set upon sending the START condition.

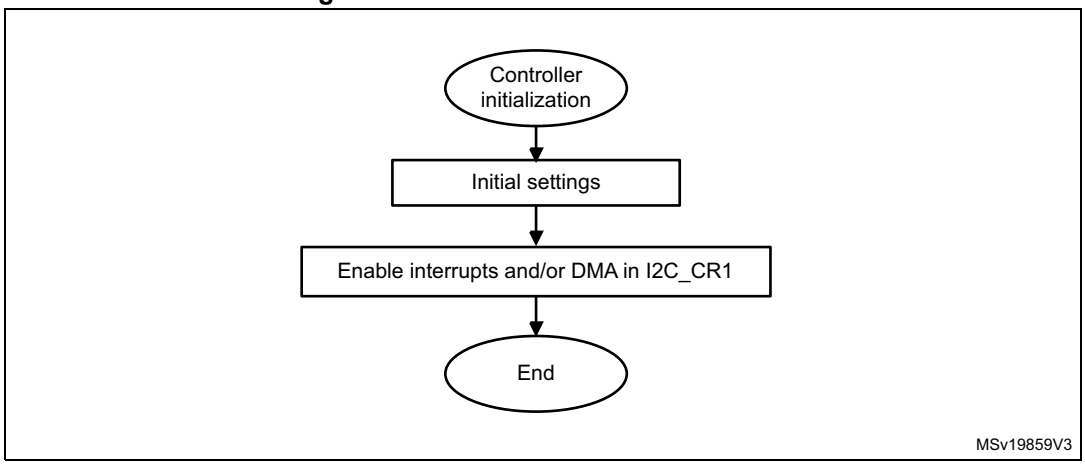
In case of an arbitration loss, the controller automatically switches back to target mode and can acknowledge its own address if it is addressed as a target.

Note: The START bit is reset by hardware when the target address is sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware upon arbitration loss.

In 10-bit addressing mode, the controller automatically keeps resending the target address in a loop until the first address byte (first seven address bits) is acknowledged by the target. Setting the ADDRCF bit makes I2C quit that loop.
If the I2C peripheral is addressed as a target (ADDR = 1) while the START bit is set, the I2C peripheral switches to target mode and the START bit is cleared.

Note: The same procedure is applied for a repeated START condition. In this case, BUSY = 1.

Figure 290. Controller initialization flow

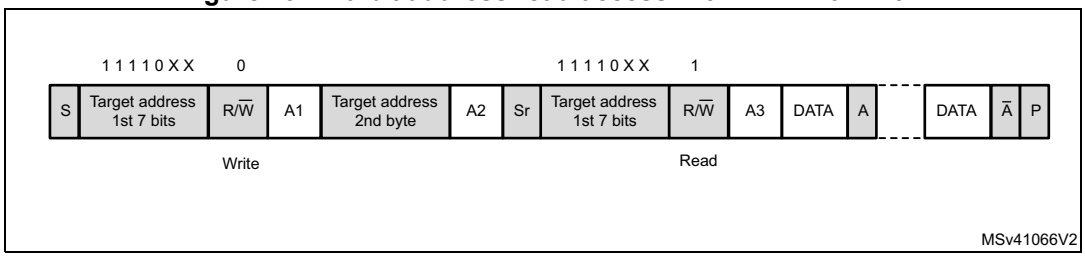


Initialization of a controller receiver addressing a 10-bit address target

If the target address is in 10-bit format, the user can choose to send the complete read sequence, by clearing the HEAD10R bit of the I2C_CR2 register. In this case, the controller automatically sends the following complete sequence after the START bit is set:

(RE)START + Target address 10-bit header Write + Target address second byte + (RE)START + Target address 10-bit header Read.

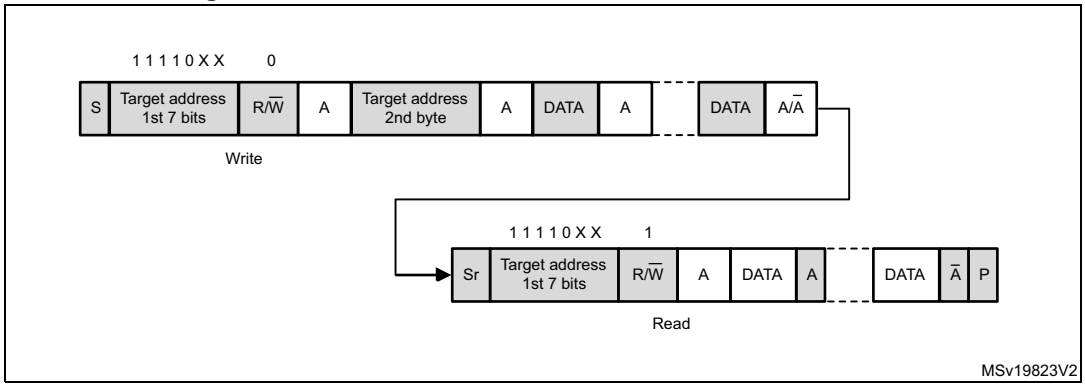
Figure 291. 10-bit address read access with HEAD10R = 0



If the controller addresses a 10-bit address target, transmits data to this target and then reads data from the same target, a controller transmission flow must be done first. Then a repeated START is set with the 10-bit target address configured with HEAD10R = 1. In this case, the controller sends this sequence:

RESTART + Target address 10-bit header Read.

Figure 292. 10-bit address read access with HEAD10R = 1



Controller transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the ninth SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit of the I2C_CR1 register is set. The flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to transmit is greater than 255, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and the number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a STOP condition is automatically sent.
- In software end mode (AUTOEND = 0), the TC flag is set and the SCL line is stretched low, to perform software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

When a NACK is received, the TXIS flag is not set and a STOP condition is automatically sent. The NACKF flag of the I2C_ISR register is set. An interrupt is generated if the NACKIE bit is set.

Figure 293. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes

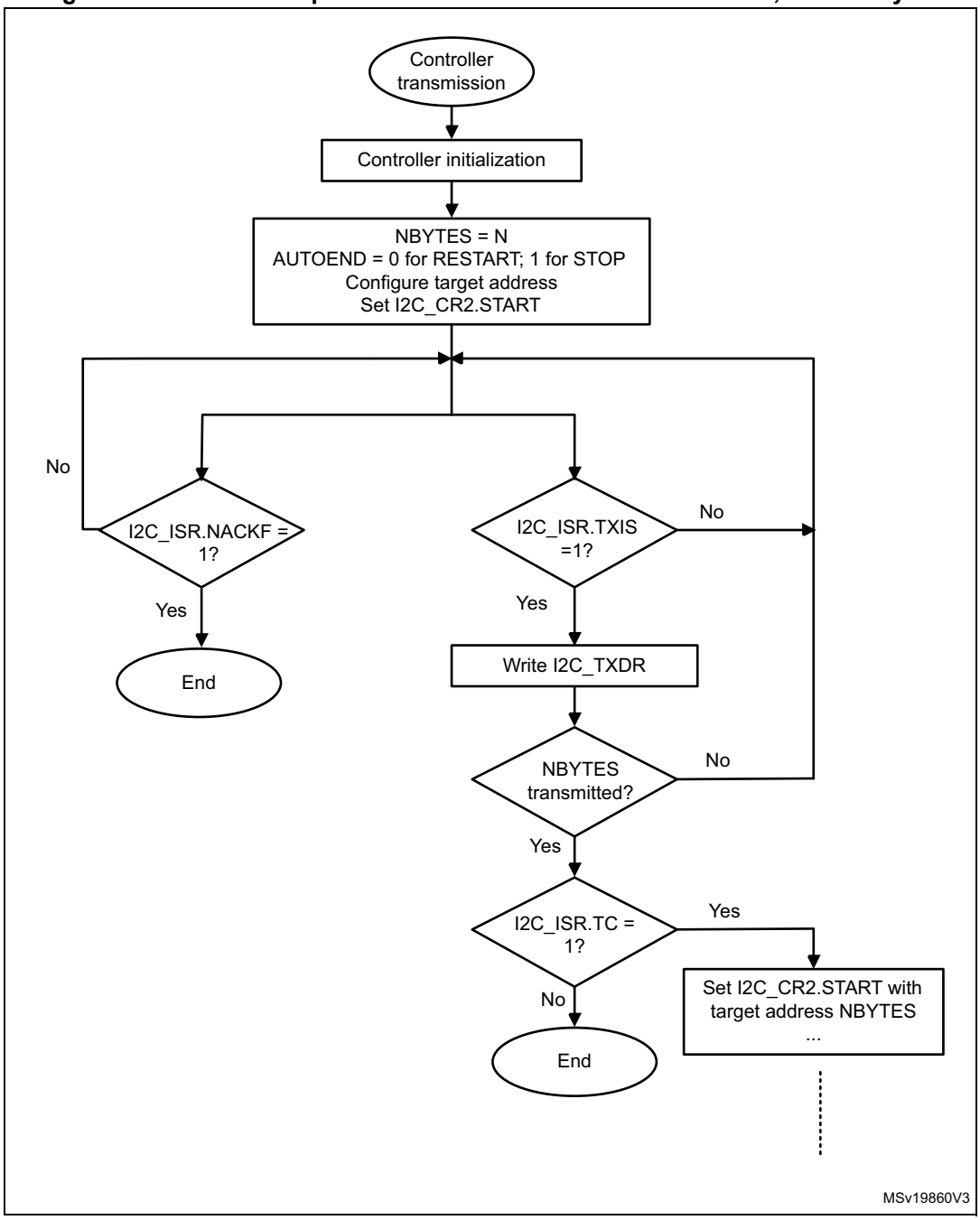


Figure 294. Transfer sequence flow for I2C controller transmitter, N > 255 bytes

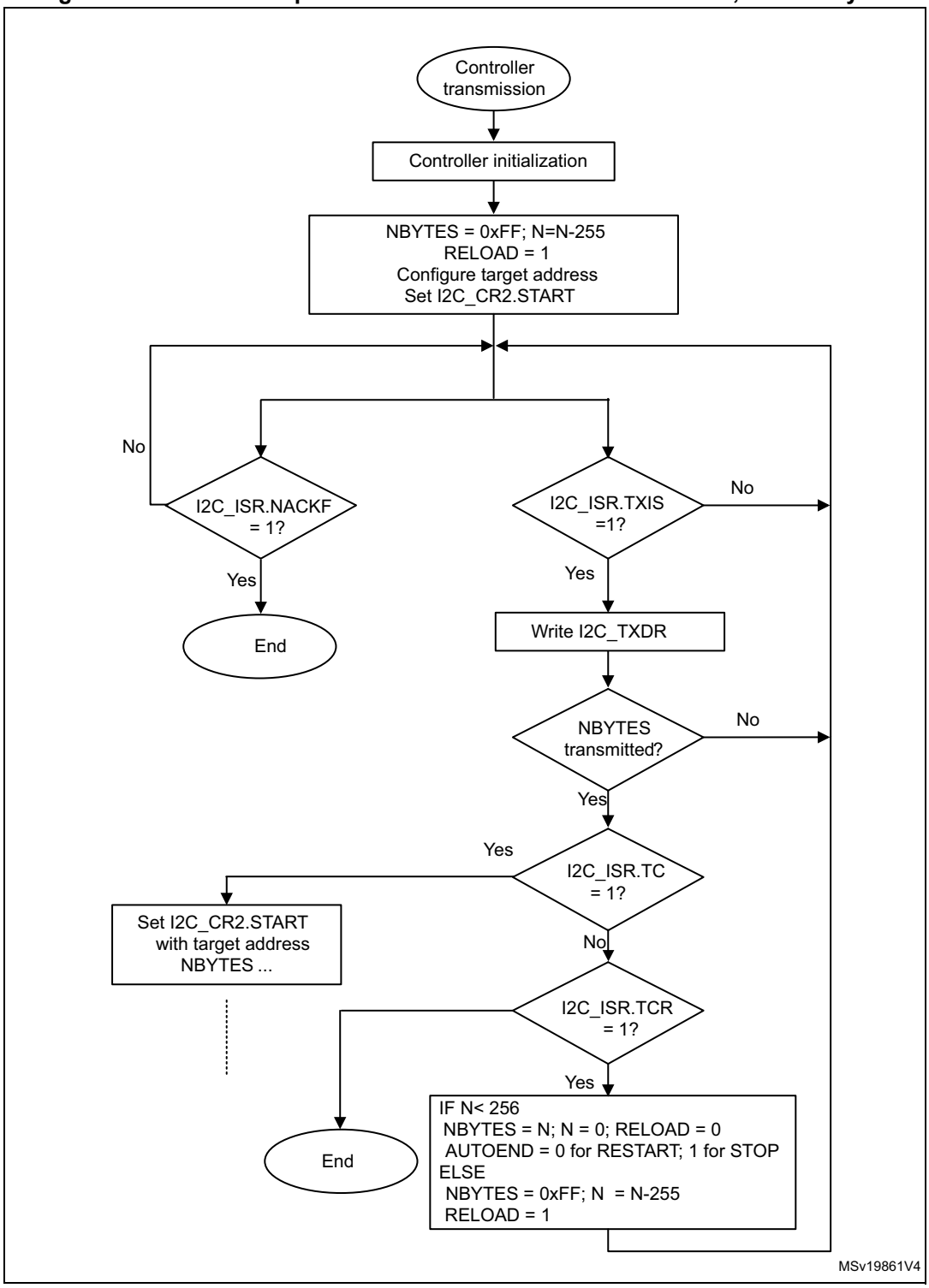
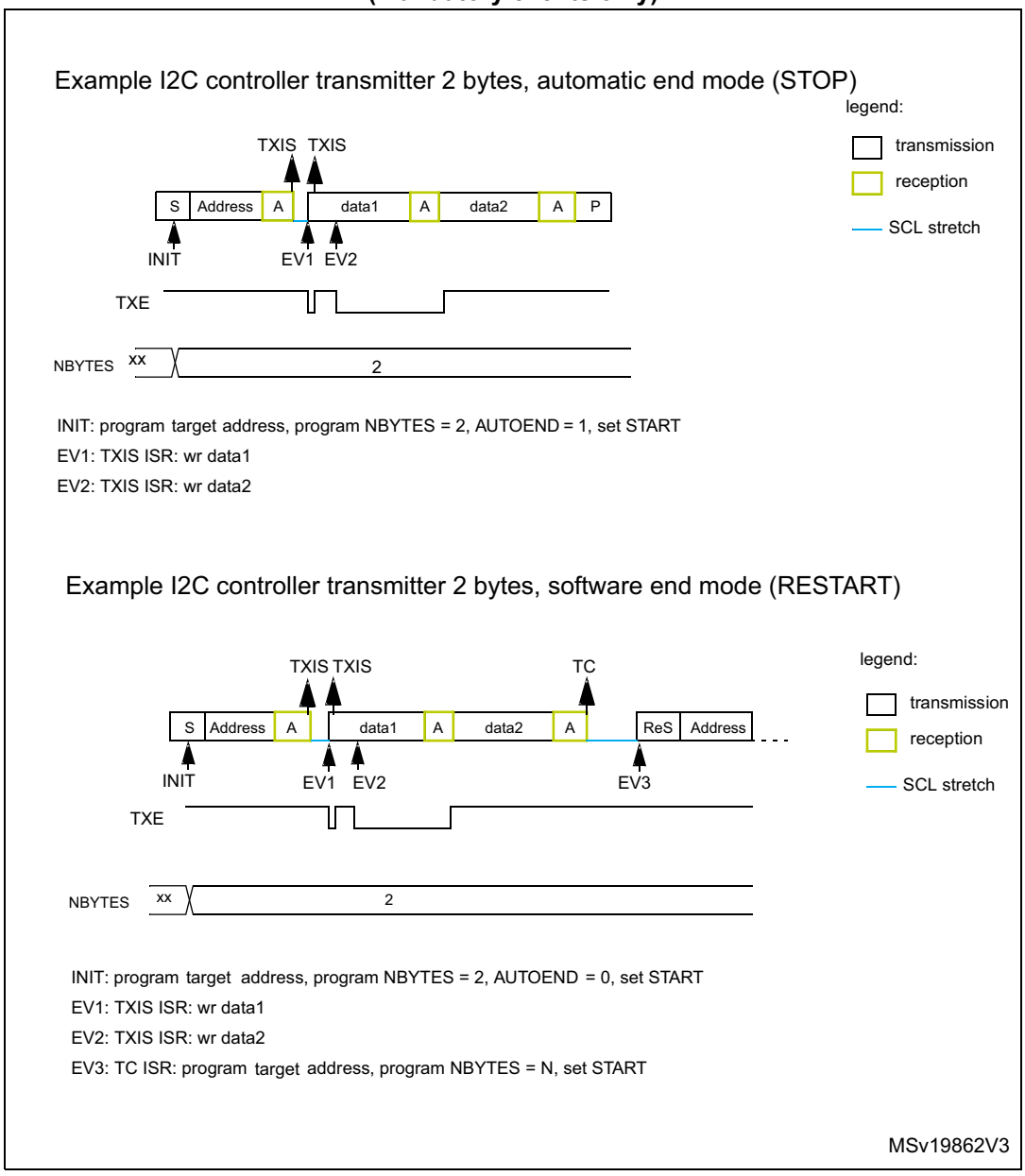


Figure 295. Transfer bus diagrams for I2C controller transmitter (mandatory events only)



Controller receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the eighth SCL pulse. An RXNE event generates an interrupt if the RXIE bit of the I2C_CR1 register is set. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to receive is greater than 255, select the reload mode, by setting the RELOAD bit of the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and he number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a NACK and a STOP are automatically sent after the last received byte.
- In software end mode (AUTOEND = 0), a NACK is automatically sent after the last received byte. The TC flag is set and the SCL line is stretched low in order to allow software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register, with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition and the target address on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

Figure 296. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes

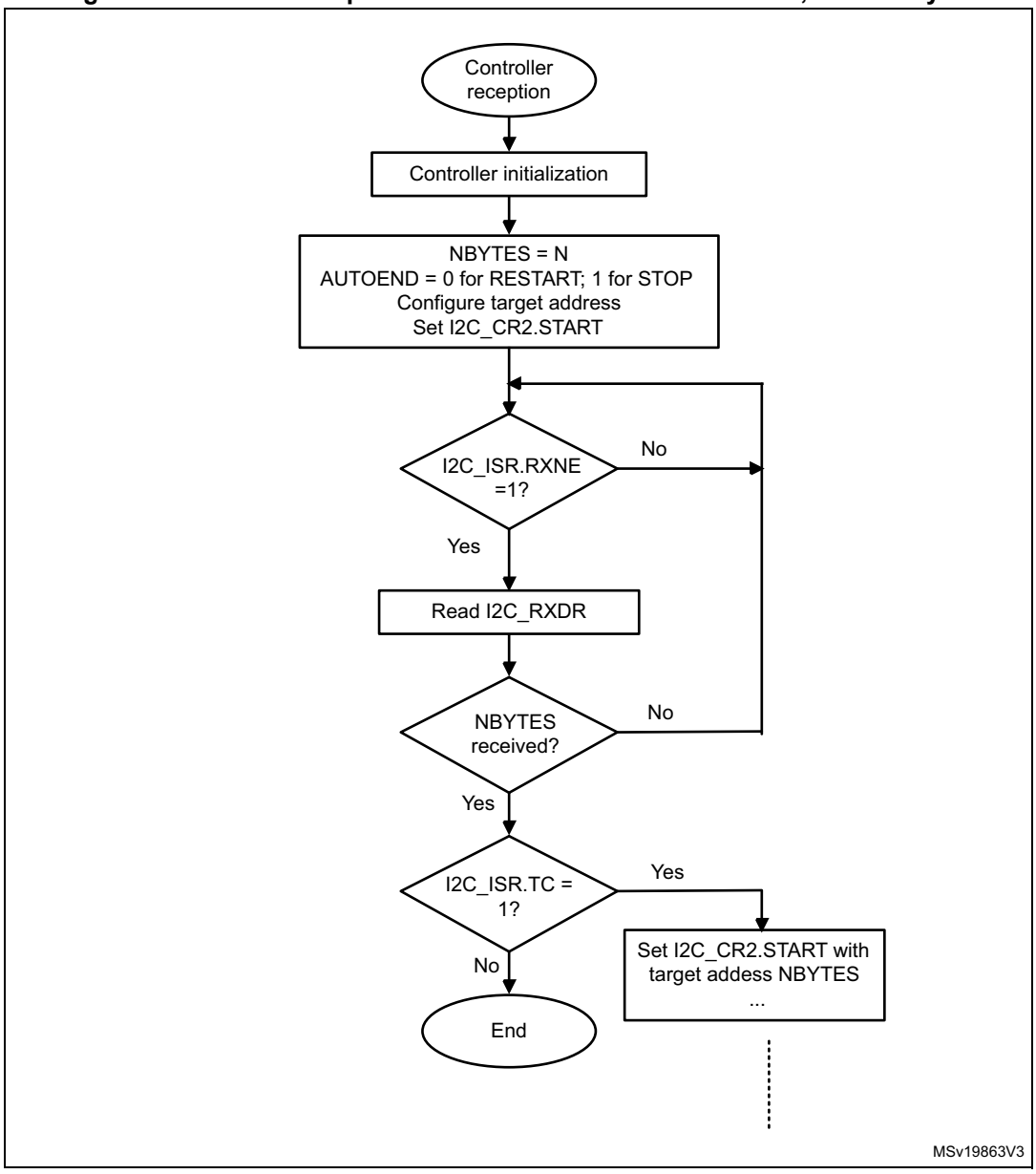


Figure 297. Transfer sequence flow for I2C controller receiver, N > 255 bytes

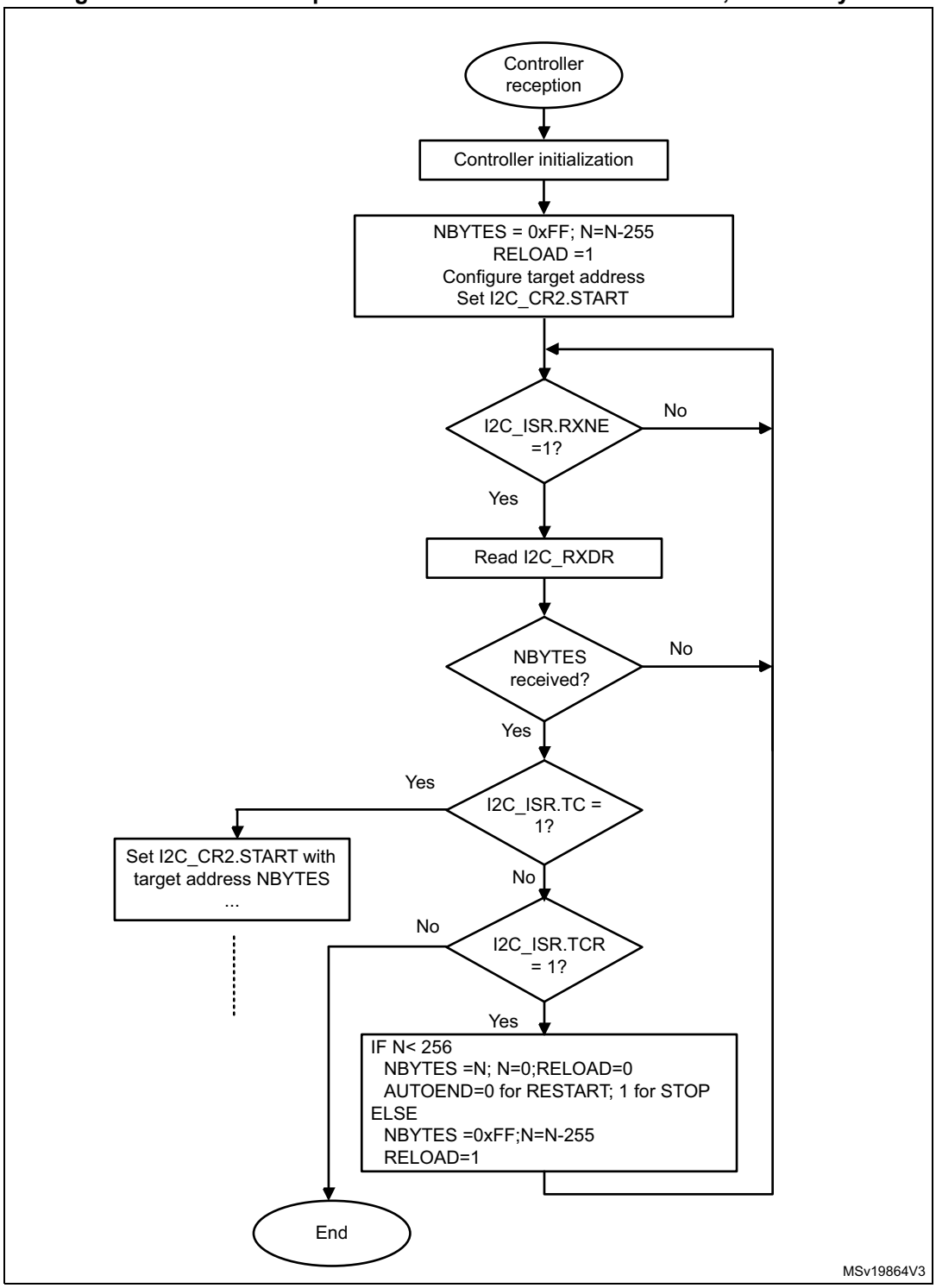
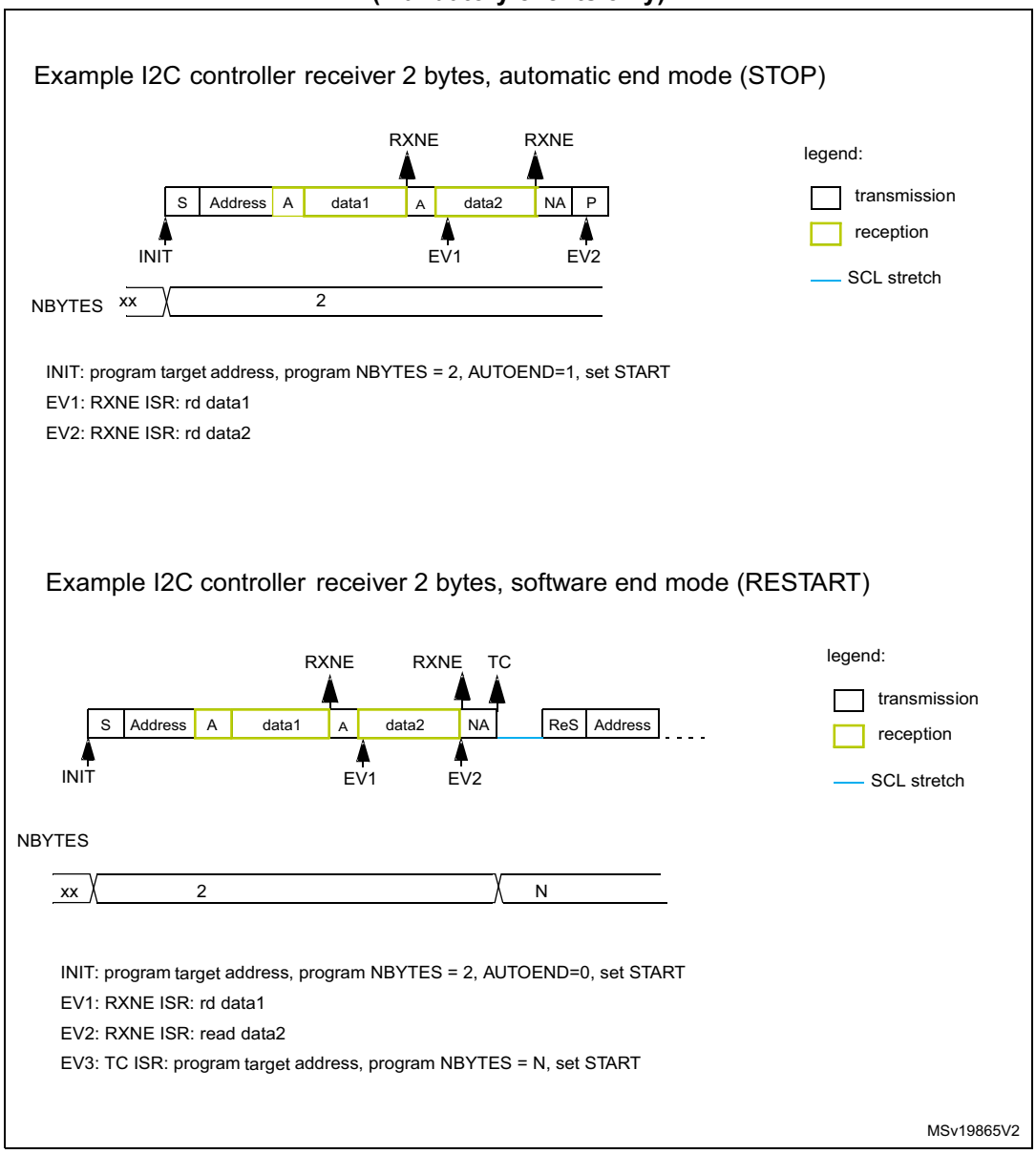


Figure 298. Transfer bus diagrams for I2C controller receiver (mandatory events only)



35.4.10 I2C_TIMINGR register configuration examples

The following tables provide examples of how to program the I2C_TIMINGR register to obtain timings compliant with the I²C-bus specification. To get more accurate configuration values, use the STM32CubeMX tool (*I2C Configuration* window).

Table 332. Timing settings for f_{I2CCLK} of 8 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC[3:0]	0x1	0x1	0x0	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x6
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	7 x 125 ns = 875 ns
SCLH[7:0]	0xC3	0xF	0x3	0x3
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	4 x 125 ns = 500 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~2.0 µs(4)
SDADEL[3:0]	0x2	0x2	0x1	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	1 x 125 ns = 125 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x1
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 655 ns.

Table 333. Timing settings for f_{I2CCLK} of 16 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC[3:0]	0x3	0x3	0x1	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x4
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH[7:0]	0xC3	0xF	0x3	0x2
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~1.0 µs(4)
SDADEL[3:0]	0x2	0x2	0x2	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x2
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 500 ns.

35.4.11 SMBus specific features

Introduction

The system management bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on operation principles of the I²C-bus. The SMBus provides a control bus for system and power management related tasks.

The I2C peripheral is compatible with the SMBus specification (http://smbus.org).

The system management bus specification refers to three types of devices:

- **Target** is a device that receives or responds to a command.
- **Controller** is a device that issues commands, generates clocks, and terminates the transfer.
- **Host** is a specialized controller that provides the main interface to the system CPU. A host must be a controller-target and must support the SMBus *host notify* protocol. Only one host is allowed in a system.

The I2C peripheral can be configured as a controller or a target device, and also as a host.

Bus protocols

There are eleven possible command protocols for any given device. The device can use any or all of them to communicate. These are: *Quick Command*, *Send Byte*, *Receive Byte*, *Write Byte*, *Write Word*, *Read Byte*, *Read Word*, *Process Call*, *Block Read*, *Block Write*, and *Block Write-Block Read Process Call*. The protocols must be implemented by the user software.

For more details on these protocols, refer to the SMBus specification (http://smbus.org).

STM32CubeMX implements an SMBus stack thanks to X-CUBE-SMBUS, a downloadable software pack that allows basic SMBus configuration per I2C instance.

Address resolution protocol (ARP)

SMBus target address conflicts can be resolved by dynamically assigning a new unique address to each target device. To provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique 128-bit device identifier (UDID). In the I2C peripheral, it is implemented by software.

The I2C peripheral supports the Address resolution protocol (ARP). The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register. The ARP commands must be implemented by the user software.

Arbitration is also performed in target mode for ARP support.

For more details on the SMBus address resolution protocol, refer to the SMBus specification (http://smbus.org).

Received command and data acknowledge control

An SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Host notify protocol

To enable the host notify protocol, set the SMBHEN bit of the I2C_CR1 register. The I2C peripheral then acknowledges the SMBus host address (0b0001 000).

When this protocol is used, the device acts as a controller and the host as a target.

SMBus alert

The I2C peripheral supports the SMBALERT# optional signal through the SMBA pin. With the SMBALERT# signal, an SMBus target device can signal to the SMBus host that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the alert response address (0b0001 100). Only the device/devices which pulled SMBALERT# low acknowledges/acknowledge the alert response address.

When the I2C peripheral is configured as an SMBus target device (SMBHEN = 0), the SMBA pin is pulled low by setting the ALERTEN bit of the I2C_CR1 register. The alert response address is enabled at the same time.

When the I2C peripheral is configured as an SMBus host (SMBHEN = 1), the ALERT flag of the I2C_ISR register is set when a falling edge is detected on the SMBA pin and ALERTEN = 1. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. When ALERTEN = 0, the alert line is considered high even if the external SMBA pin is low.

Note: If the SMBus alert pin is not required, keep the ALERTEN bit cleared. The SMBA pin can then be used as a standard GPIO.

Packet error checking

A packet error checking mechanism introduced in the SMBus specification improves reliability and communication robustness. The packet error checking is implemented by appending a packet error code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x^8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The I2C peripheral embeds a hardware PEC calculator and allows a not acknowledge to be sent automatically when the received byte does not match the hardware calculated PEC.

Timeouts

To comply with the SMBus timeout specifications, the I2C peripheral embeds hardware timers.

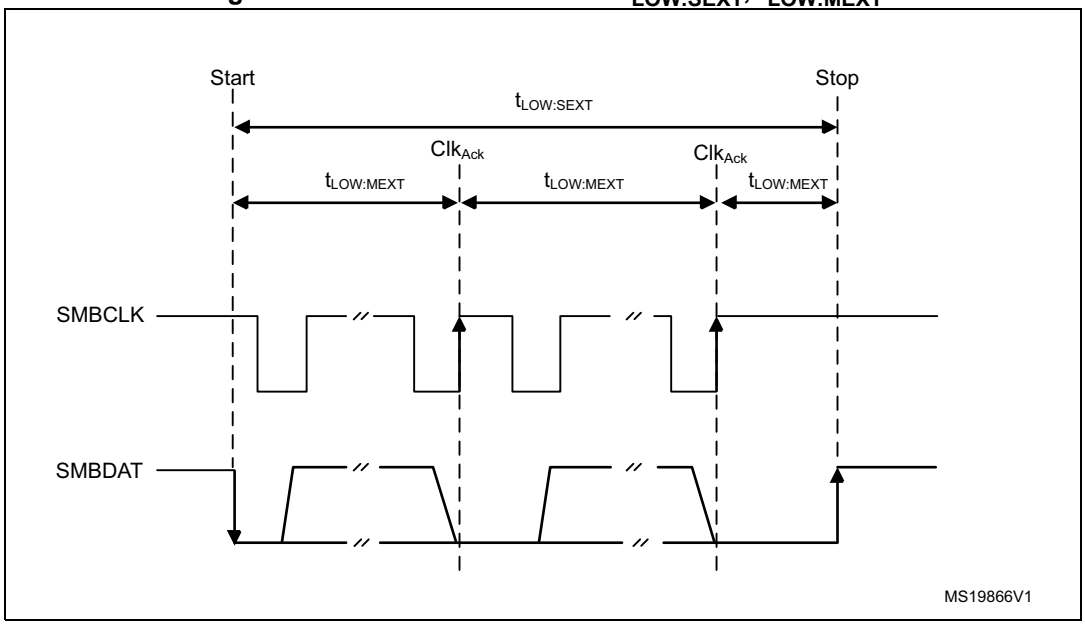
Table 334. SMBus timeout specifications

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
$t_{LOW:SEXT}$(1)	Cumulative clock low extend time (target device)	-	25	
$t_{LOW:MEXT}$(2)	Cumulative clock low extend time (controller device)	-	10	

1. $t_{LOW:SEXT}$ is the cumulative time a given target device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that another target device or the controller also extends the clock causing the combined clock low extend time to be greater than $t_{LOW:SEXT}$. The value provided applies to a single target device connected to a full-target controller.

2. $t_{LOW:MEXT}$ is the cumulative time a controller device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a target device or another controller also extends the clock, causing the combined clock low time to be greater than $t_{LOW:MEXT}$ on a given byte. The value provided applies to a single target device connected to a full-target controller.

Figure 299. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$



Bus idle detection

A controller can assume that the bus is free if it detects that the clock and data signals have been high for $t_{IDLE} > t_{HIGH}$(max) (refer to the table in *Section 35.4.9*).

This timing parameter covers the condition where a controller is dynamically added to the bus, and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the controller must wait long enough to ensure that a transfer is not currently in progress. The I2C peripheral supports a hardware bus idle detection.

35.4.12 SMBus initialization

In addition to the I2C initialization for the I²C-bus, the use of the peripheral for the SMBus communication requires some extra initialization steps.

Received command and data acknowledge control (target mode)

An SMBus receiver must be able to NACK each received command or data. To allow ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Specific addresses (target mode)

The specific SMBus addresses must be enabled if required. Refer to *Bus idle detection* for more details.

The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register.

The SMBus host address (0b0001 000) is enabled by setting the SMBHEN bit of the I2C_CR1 register.

The alert response address (0b0001100) is enabled by setting the ALERTEN bit of the I2C_CR1 register.

Packet error checking

PEC calculation is enabled by setting the PECEN bit of the I2C_CR1 register. Then the PEC transfer is managed with the help of the hardware byte counter associated with the NBYTES[7:0] bitfield of the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in target mode. The PEC is transferred after transferring NBYTES[7:0] - 1 data bytes, if the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

Caution: Changing the PECEN configuration is not allowed when the I2C peripheral is enabled.

Table 335. SMBus with PEC configuration

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Controller Tx/Rx NBYTES + PEC+ STOP	X	0	1	1
Controller Tx/Rx NBYTES + PEC + ReSTART	X	0	0	1
Target Tx/Rx with PEC	1	0	X	1

Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits of the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

$t_{TIMEOUT}$ check

To check the $t_{TIMEOUT}$ parameter, load the 12-bit TIMEOUTA[11:0] bitfield with the timer reload value. Keep the TIDLE bit at 0 to detect the SCL low level timeout.

Then set the TIMOUTEN bit of the I2C_TIMEOUTR register, to enable the timer.

If SCL is tied low for longer than the (TIMEOUTA + 1) x 2048 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 336*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

$t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ check

A 12-bit timer associated with the TIMEOUTB[11:0] bitfield allows checking $t_{LOW:SEXT}$ for the I2C peripheral operating as a target, or $t_{LOW:MEXT}$ when it operates as a controller. As the standard only specifies a maximum, the user can choose the same value for both. The timer is then enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

If the SMBus peripheral performs a cumulative SCL stretch for longer than the (TIMEOUTB + 1) x 2048 x t_{I2CCLK} period, and within the timeout interval described in *Bus idle detection* section, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 337*.

Caution: Changing the TIMEOUTB[11:0] bitfield value is not allowed when the TEXTEN bit is set.

Bus idle detection

To check the t_{IDLE} period, the TIMEOUTA[11:0] bitfield associated with 12-bit timer must be loaded with the timer reload value. Keep the TIDLE bit at 1 to detect both SCL and SDA high level timeout. Then set the TIMOUTEN bit of the I2C_TIMEOUTR register to enable the timer.

If both the SCL and SDA lines remain high for longer than the (TIMEOUTA + 1) x 4 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 338*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

35.4.13 SMBus I2C_TIMEOUTR register configuration examples

The following tables provide examples of settings to reach desired $t_{TIMEOUT}$, $t_{LOW:SEXT}$, $t_{LOW:MEXT}$, and t_{IDLE} timings at different f_{I2CCLK} frequencies.

Table 336. TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	$t_{TIMEOUT}$
8 MHz	0x61	0	1	98 x 2048 x 125 ns = 25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms

Table 337. TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms

f_{I2CCLK}	TIMEOUTB[11:0]	TEXTEN	$t_{LOW:SEXT}$ $t_{LOW:MEXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms

Table 338. TIMEOUTA[11:0] for maximum t_{IDLE} of 50 µs

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	t_{IDLE}
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 µs
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 µs

35.4.14 SMBus target mode

In addition to I2C target transfer management (refer to *Section 35.4.8: I2C target mode*), this section provides extra software flowcharts to support SMBus.

SMBus target transmitter

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case, the total number of TXIS interrupts is NBYTES[7:0] - 1, and the content of the I2C_PECR register is automatically transmitted if the controller requests an extra byte after the transfer of the NBYTES[7:0] - 1 data bytes.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 300. Transfer sequence flow for SMBus target transmitter N bytes + PEC

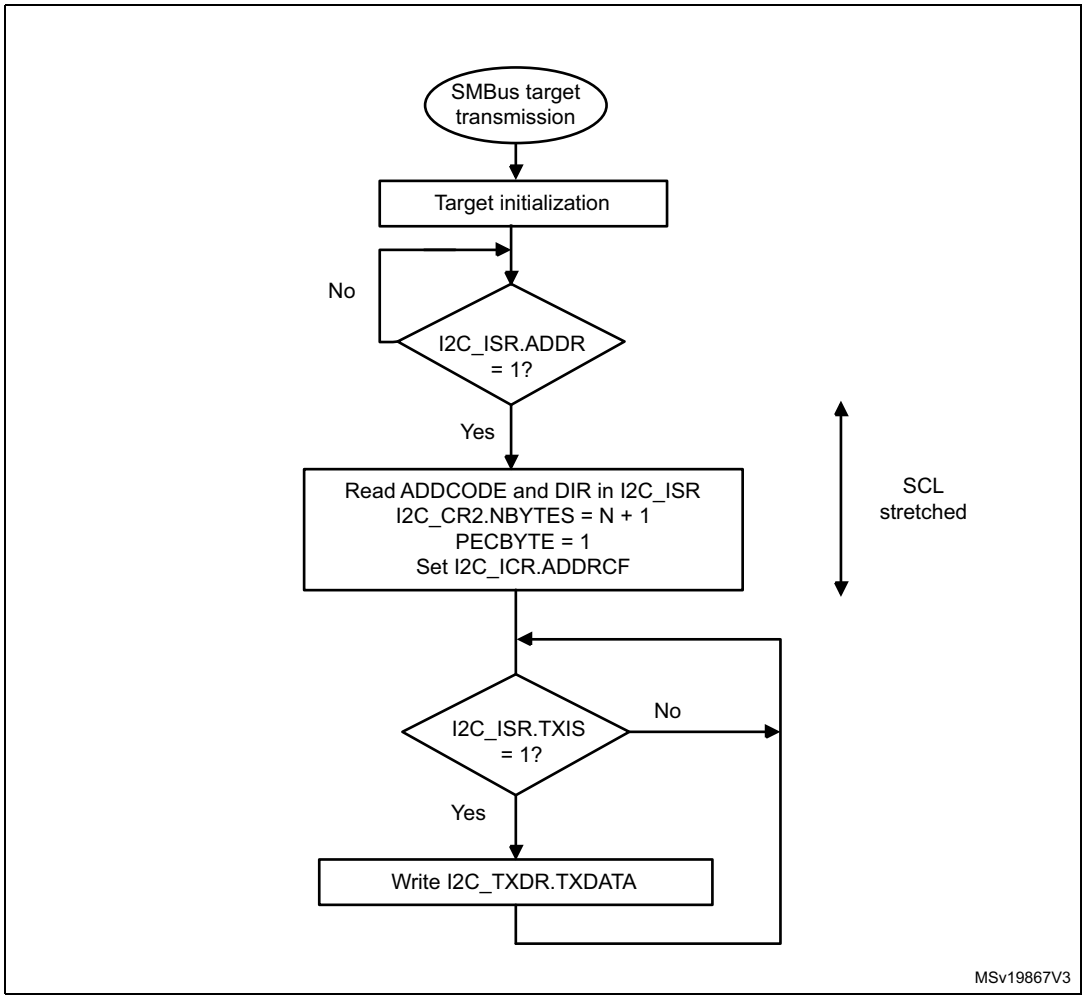
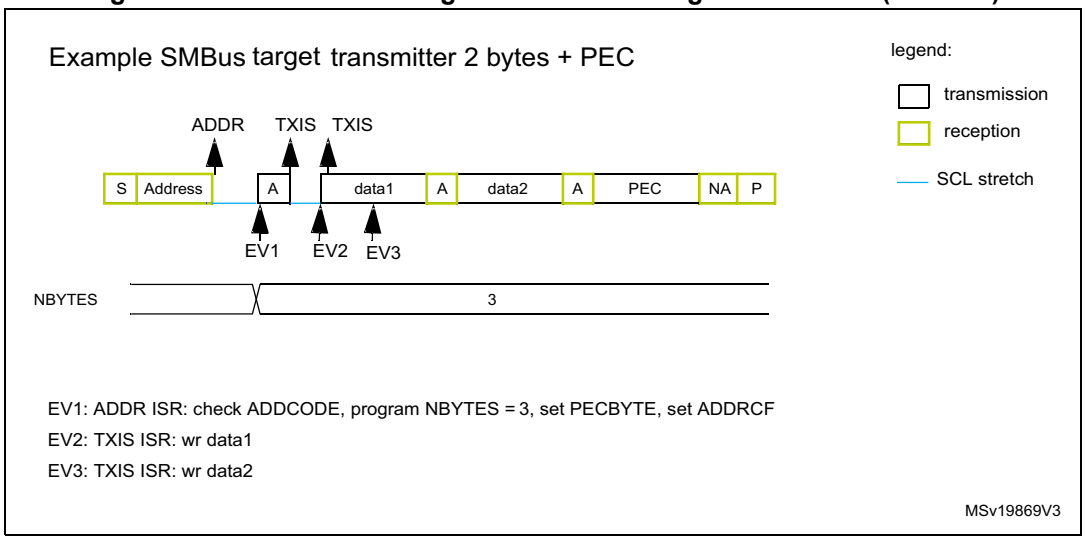


Figure 301. Transfer bus diagram for SMBus target transmitter (SBC = 1)



SMBus target receiver

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC checking at the end of the programmed number of data bytes. To allow the ACK control of each byte, the reload mode must be selected (RELOAD = 1). Refer to *Target byte control mode* for more details.

To check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

Upon a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

If no ACK software control is required, the user can set the PECBYTE bit and, in the same write operation, load NBYTES[7:0] with the number of bytes to receive in a continuous flow. After the receipt of NBYTES[7:0] - 1 bytes, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 302. Transfer sequence flow for SMBus target receiver N bytes + PEC

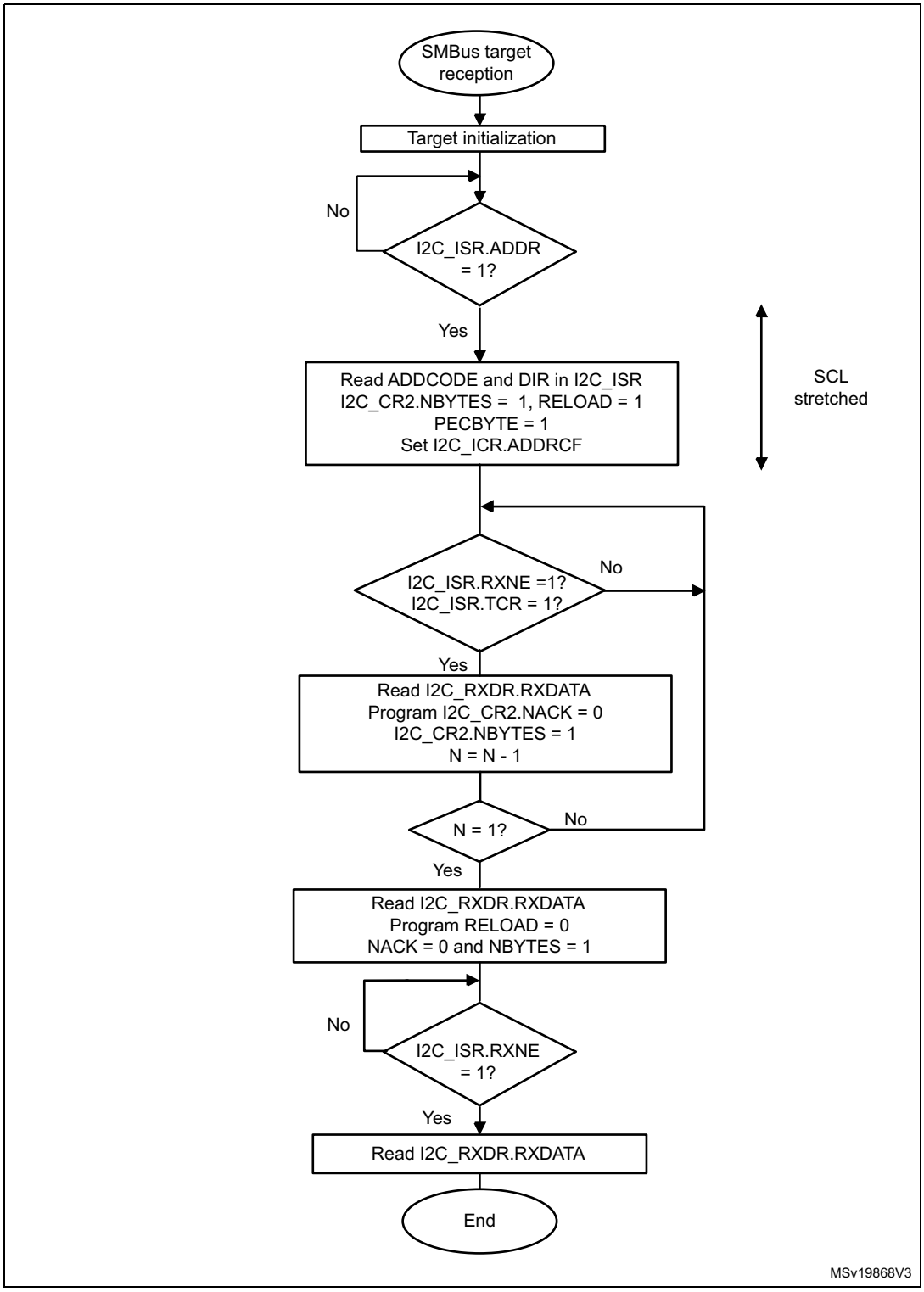
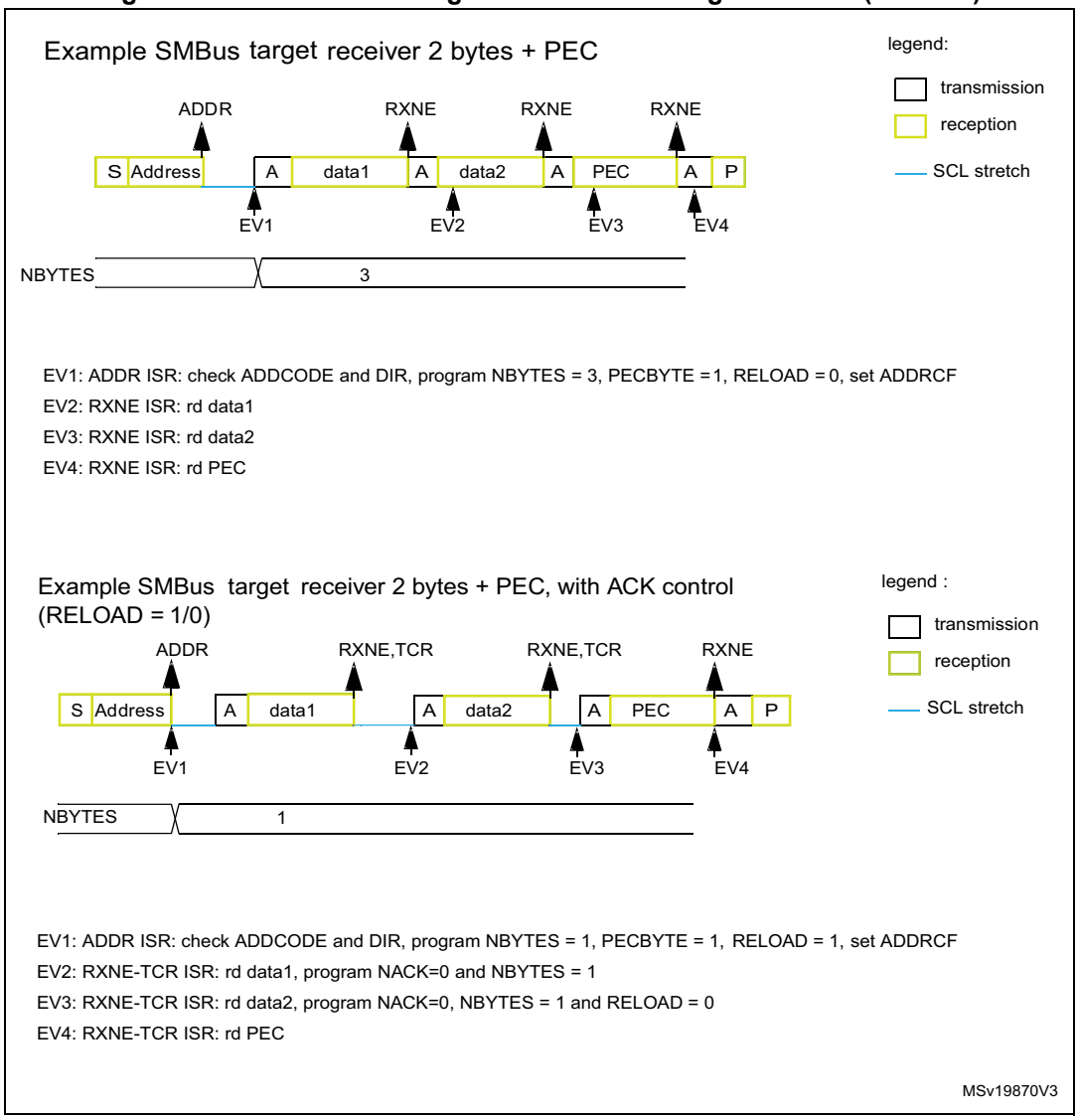


Figure 303. Bus transfer diagrams for SMBus target receiver (SBC = 1)



35.4.15 SMBus controller mode

In addition to I2C controller transfer management (refer to *Section 35.4.9: I2C controller mode*), this section provides extra software flowcharts to support SMBus.

SMBus controller transmitter

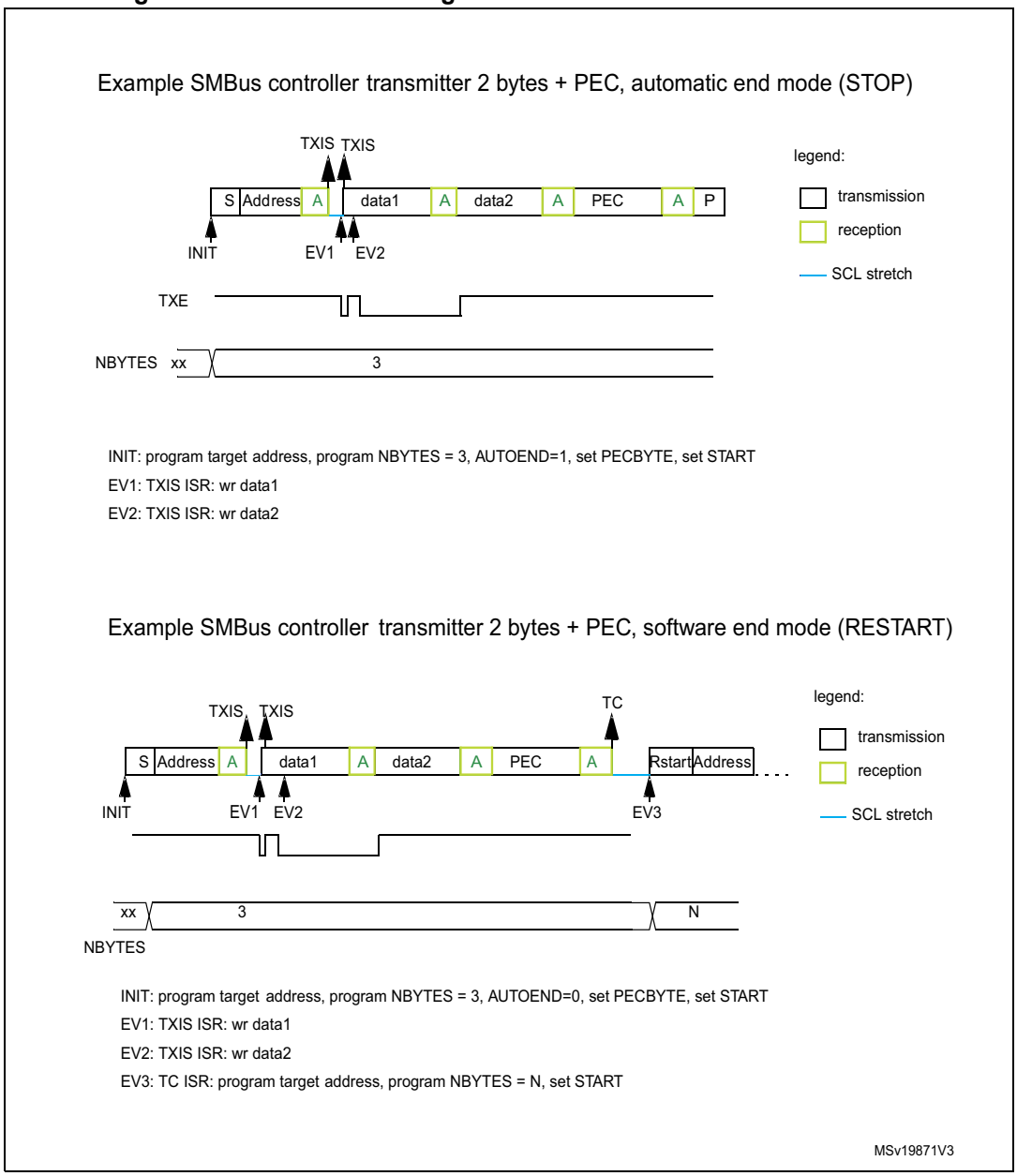
When the SMBus controller wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be loaded in the NBYTES[7:0] bitfield, before setting the START bit. In this case, the total number of TXIS interrupts is NBYTES[7:0] - 1. So if the PECBYTE bit is set when NBYTES[7:0] = 0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus controller wants to send a STOP condition after the PEC, the automatic end mode must be selected (AUTOEND = 1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus controller wants to send a RESTART condition after the PEC, the software mode must be selected (AUTOEND = 0). In this case, once NBYTES[7:0] - 1 are transmitted, the I2C_PECR register content is transmitted. The TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 304. Bus transfer diagrams for SMBus controller transmitter



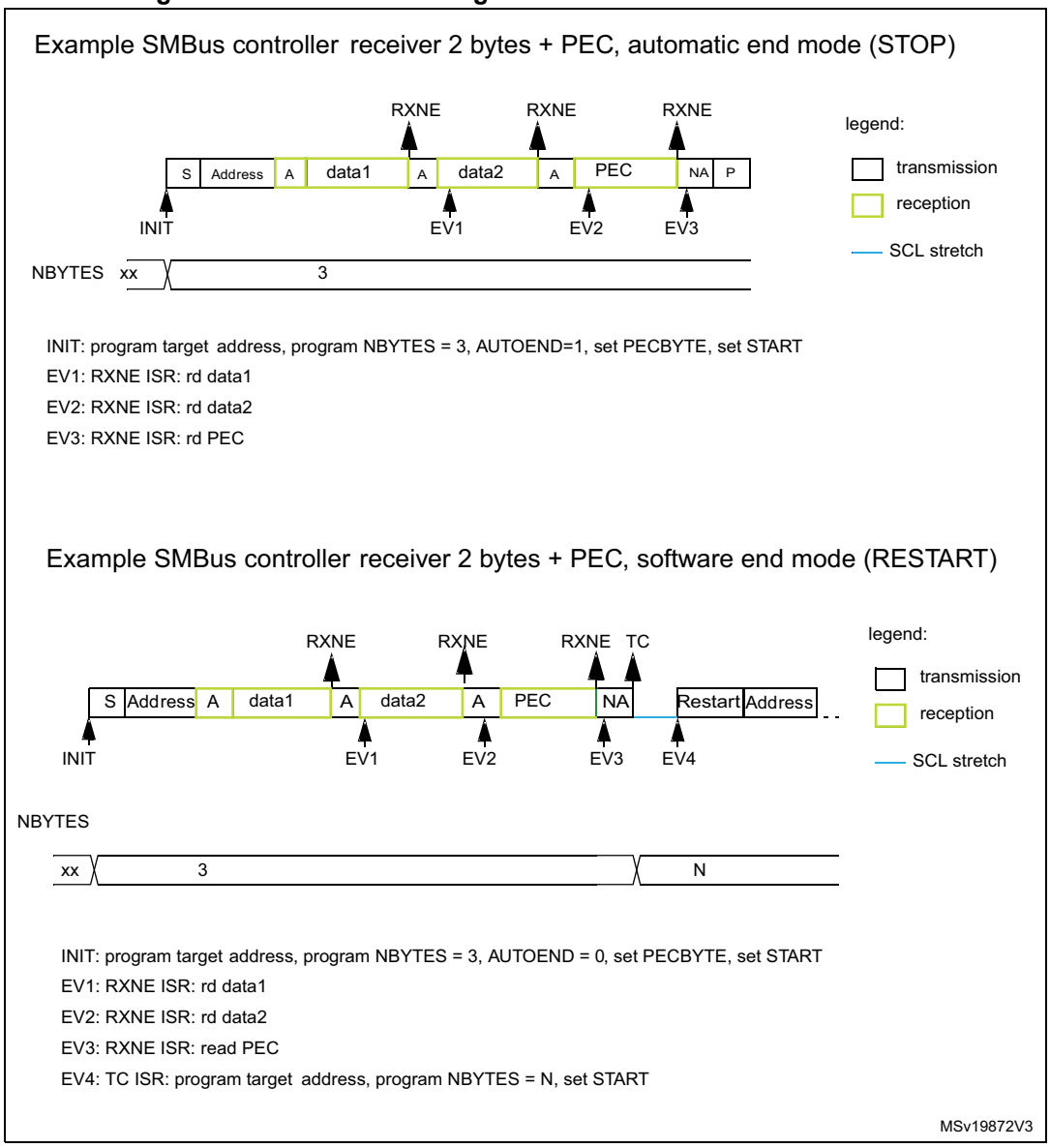
SMBus controller receiver

When the SMBus controller wants to receive, at the end of the transfer, the PEC followed by a STOP condition, the automatic end mode can be selected (AUTOEND = 1). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus controller receiver wants to receive, at the end of the transfer, the PEC byte followed by a RESTART condition, the software mode must be selected (AUTOEND = 0). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 305. Bus transfer diagrams for SMBus controller receiver



35.4.16 Autonomous mode

The I2C peripheral can autonomously operate as controller or target in Stop mode as long as NOSTRETCH = 0. The autonomous mode is not supported when NOSTRETCH = 1. Besides Stop mode, the autonomous mode can also be used in Run and Sleep modes.

The APB clock is requested by the peripheral each time the I2C status needs to be updated. Once the APB clock is received by the peripheral, either an interrupt or a DMA request is generated, depending on the I2C configuration.

If an interrupt is generated, the device wakes up from Stop mode.

If there is no interrupt, the device remains in Stop mode, but the kernel and AHB/APB clocks are available for the I2C peripheral and all the autonomous peripherals enabled in the RCC.

If the DMA requests are enabled, the data are DMA-transferred to or from the SRAM, while the device remains in Stop mode.

Target mode

In target mode, the autonomous mode is enabled in Stop mode if WUPEN = 1 in the I2C_CR1. This mode is supported only when NOSTRETCH = 0.

The kernel clock is requested by the peripheral on START detection during all the transfer until the STOP condition occurs, when the target is addressed. If the target is not addressed, the kernel clock request is released after the address phase.

To optimize the functionality in target mode by using only DMA transfers, it is possible to set the ADDRACLR and STOPFACLR bits in the I2C_CR1 in order to avoid to serve the Address match (ADDR) and STOP detection (STOPF) events.

Note: If the I2C clock is the system clock, or if WUPEN = 0, the internal oscillator is not switched on after a START is received in target mode.

Controller mode

In controller mode, a transfer can be automatically launched when an asynchronous trigger is detected in Run, Sleep, or Stop mode. The trigger, selected by TRIGSEL in the I2C_AUTOCR register, generates a kernel clock request to allow the transfer, and launches a START condition and the I2C transfer as defined in the I2C_CR2 register. The kernel clock is requested until the STOP condition occurs.

To avoid waking up the CPU too often, it is possible to replace some interrupts by DMA requests, to make some control register write actions. In controller mode, *transfer complete* (TC) and *transfer complete reload* (TCR) events can generate a DMA request when the corresponding TCDMAEN or TCRDMAEN bits of the I2C_AUTOCR register are set. Consequently, the I2C_CR2 register can be written thanks to a DMA transfer, to program a new transfer (in TC event) or to reload the number of bytes (in TCR event).

In case a trigger is enabled, but the application needs to take the control back to start a different transfer, the following steps are required before writing in the I2C_CR2 register to launch the new transfer:

1. Wait for BUSY = 0 in the I2C_ISR register
2. Clear the TRIGEN bit of the I2C_AUTOCR register
3. Wait for longer than t_{BUF} (bus free time between a STOP and a START condition)
4. Wait for BUSY = 0 in the I2C_ISR register

Caution: When the device is in Stop mode, the I2C peripheral receives its kernel clock only when it is implicated in the transfer. Consequently, features such as timeouts and bus idle detection are not reliable in Stop mode.

Caution: The digital filter is not compatible with the functionality in Stop mode. Before entering Stop mode with the WUPEN bit set, deactivate the digital filter, by writing zero to the DNF[3:0] bitfield.

Caution: Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the wake-up from Stop mode feature.

Caution: If wake-up from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be disabled before entering Stop mode (PE = 0).

35.4.17 Error conditions

The following errors are the conditions that can cause the communication to fail.

Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of nine SCL clock pulses. START or STOP condition is detected when an SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C peripheral is involved in the transfer as controller or addressed target (that is, not during the address phase in target mode).

In case of a misplaced START or RESTART detection in target mode, the I2C peripheral enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag of the I2C_ISR register is set, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Arbitration loss (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

In controller mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the controller switches automatically to target mode.

In target mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Overrun/underrun error (OVR)

An overrun or underrun error is detected in target mode when NOSTRETCH = 1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF = 1 and the first data byte must be sent. The content of the I2C_TXDR register is sent if TXE = 0, 0xFF if not.
  - When a new byte must be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Packet error checking error (PECERR)

A PEC error is detected when the received PEC byte does not match the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Timeout error (TIMEOUT)

A timeout error occurs for any of these conditions:

- TIDLE = 0 and SCL remains low for the time defined in the TIMEOUTA[11:0] bitfield: this is used to detect an SMBus timeout.
- TIDLE = 1 and both SDA and SCL remains high for the time defined in the TIMEOUTA [11:0] bitfield: this is used to detect a bus idle condition.
- Controller cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:MEXT}$ parameter).
- Target cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:SEXT}$ parameter).

When a timeout violation is detected in controller mode, a STOP condition is automatically sent.

When a timeout violation is detected in target mode, the SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Alert (ALERT)

The ALERT flag is set when the I2C peripheral is configured as a host (SMBHEN = 1), the SMBALERT# signal detection is enabled (ALERTEN = 1), and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

35.5 I2C in low-power modes

Table 339. Effect of low-power modes to I2C

Mode	I2Cx	Description
Sleep	1, 3	No effect. I2C interrupts cause the device to exit this mode.
Stop 0 and Stop 1	1, 3	The contents of the I2C registers are kept. If the autonomous mode is enabled and I2C is clocked by an internal oscillator available in these modes, transfers in controller and in target modes are functional. DMA requests are functional, and the interrupts cause the device to exit these modes. If WUPEN = 0, the I2C peripheral must be disabled before entering these modes.
Stop 2	1	The peripheral is powered down and must be reinitialized after exiting this mode.
	3	The contents of the I2C registers are kept. If WUPEN = 1 and I2C is clocked by an internal oscillator (HSI16), the address recognition is functional. The I2C address match condition causes the device to exit this mode. If WUPEN = 0, the I2C peripheral must be disabled before entering this mode.
Stop 3	1	The peripheral is powered down and must be reinitialized after exiting this mode.
	3	The contents of the I2C registers are kept. Autonomous mode is not supported.
Standby	1, 3	The I2C peripheral is powered down. It must be reinitialized after exiting this mode.

35.6 I2C interrupts

The following table gives the list of I2C interrupt requests.

Table 340. I2C interrupt requests

Interrupt acronym		Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit the Sleep mode	Exit the Stop mode	Exit the Standby mode
I2C	I2C_EV	Receive buffer not empty	RXNE	RXIE	Read I2C_RXDR register	Yes	Yes(1)	No
		Transmit buffer interrupt status	TXIS	TXIE	Write I2C_TXDR register			
		STOP detection interrupt flag	STOPF	STOPIE	Write STOPCF = 1			
		Transfer complete reload	TCR	TCIE	Write I2C_CR2 with NBYTES[7:0] = 0			
		Transfer complete	TC		Write START = 1 or STOP = 1			
		Address matched	ADDR	ADDRIE	Write ADDRCF=1			
		NACK reception	NACKF	NACKIE	Write NACKCF=1			
		Address matched	ADDR	ADDRIE	Write ADDRCF=1			
		NACK reception	NACKF	NACKIE	Write NACKCF=1			
	I2C_ERR	Bus error	BERR	ERRIE	Write BERRCF=1	Yes	Yes(1)	No
		Arbitration loss	ARLO		Write ARLOCF=1			
		Overrun/underrun	OVR		Write OVRCF=1			
		PEC error	PECERR		Write PECERRCF=1			
		Timeout/ t_{LOW} error	TIMEOUT		Write TIMEOUTCF=1			
		SMBus alert	ALERT		Write ALERTCF=1			

1. Refer to the I2C implementation table for information about wake-up from Stop mode support per instance.

35.7 I2C DMA requests

35.7.1 Transmission using DMA

DMA (direct memory access) can be enabled for transmission by setting the TXDMAEN bit of the I2C_CR1 register. Data is loaded from an SRAM area configured through the DMA peripheral (see *Section 16: Low-power direct memory access controller (LPDMA)*) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software (the transmitted target address cannot be transferred with DMA). When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to *Controller transmitter*.

In target mode:

- With NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
- With NOSTRETCH = 1, the DMA must be initialized before the address match event.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target transmitter* and *SMBus controller transmitter*.

Note: If DMA is used for transmission, it is not required to set the TXIE bit.

35.7.2 Reception using DMA

DMA (direct memory access) can be enabled for reception by setting the RXDMAEN bit of the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured through the DMA peripheral (refer to *Section 16: Low-power direct memory access controller (LPDMA)*) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.

In target mode with NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target receiver* and *SMBus controller receiver*.

Note: If DMA is used for reception, it is not required to set the RXIE bit.

35.7.3 Controller event control using DMA

In controller mode, the transfer can be automatically managed while the device is in Run, Sleep, or Stop mode, thanks to TC and TCR DMA requests.

If the TCDMAEN bit of the I2C_AUTOCR register is set, the I2C_EVC (I2C control event) DMA request is generated when the TC bit of the I2C_ISR register is set. The DMA controller must be programmed to write the next command in the I2C_CR2 register. If both STOP and START bits are set in the new I2C_CR2 command, a STOP condition followed by a START condition is sent, followed by the address, direction and number of bytes defined in I2C_CR2. If only START bit is set in the new I2C_CR2 command, a RESTART condition is sent, followed by the address, direction and number of bytes defined in I2C_CR2.

If the TCRDMAEN bit of the I2C_AUTOCR register is set, the I2C_EVC (I2C control event) DMA request is generated when TCR bit of the I2C_ISR register is set. The DMA controller must be programmed to write the remaining number of bytes to transfer in the I2C_CR2 register.

35.8 I2C debug modes

When the device enters debug mode (core halted), the SMBus timeout either continues working normally or stops, depending on the DBG_I2Cx_STOP bits in the DBG block.

35.9 I2C registers

Refer to *Section 1.2* for the list of abbreviations used in register descriptions.

The registers are accessed by words (32-bit).

35.9.1 I2C control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STOPFACLR	ADDRACLR	Res.	Res.	Res.	Res.	Res.	FMP	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC
rw	rw						rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **STOPFACLR**: STOP detection flag (STOPF) automatic clear
0: STOPF flag is set by hardware, cleared by software by setting STOPCF bit.
1: STOPF flag remains cleared by hardware. This mode can be used in NOSTRETCH target mode, to avoid the overrun error if the STOPF flag is not cleared before next data transmission. This allows a target data management by DMA only, without any interrupt from peripheral.

Bit 30 **ADDRACLR**: Address match flag (ADDR) automatic clear
0: ADDR flag is set by hardware, cleared by software by setting ADDRCF bit.
1: ADDR flag remains cleared by hardware. This mode can be used in target mode, to avoid the ADDR clock stretching if the I2C enables only one target address. This allows a target data management by DMA only, without any interrupt from peripheral.

Bits 29:25 Reserved, must be kept at reset value.

Bit 24 **FMP**: Fast-mode Plus drive enable
0: FMP I/O drive disabled
1: FMP I/O drive enabled

Bit 23 **PECEN:** PEC enable
0: PEC calculation disabled
1: PEC calculation enabled

Bit 22 **ALERTEN**: SMBus alert enable

0: The SMBALERT# signal on SMBA pin is not supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is released and the alert response address header is disabled (0001100x followed by NACK).
1: The SMBALERT# signal on SMBA pin is supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is driven low and the alert response address header is enabled (0001100x followed by ACK).

Note: When ALERTEN = 0, the SMBA pin can be used as a standard GPIO.

Bit 21 **SMBDEN**: SMBus device default address enable

0: Device default address disabled. Address 0b1100001x is NACKed.
1: Device default address enabled. Address 0b1100001x is ACKed.

Bit 20 **SMBHEN**: SMBus host address enable

0: Host address disabled. Address 0b0001000x is NACKed.
1: Host address enabled. Address 0b0001000x is ACKed.

Bit 19 **GCEN**: General call enable

0: General call disabled. Address 0b00000000 is NACKed.
1: General call enabled. Address 0b00000000 is ACKed.

Bit 18 **WUPEN**: Wake-up from Stop mode enable

0: Wake-up from Stop mode disabled.
1: Wake-up from Stop mode enabled.

Note: WUPEN can be set only when DNF[3:0] = 0000.

Bit 17 **NOSTRETCH**: Clock stretching disable

This bit is used to disable clock stretching in target mode. It must be kept cleared in controller mode.
0: Clock stretching enabled
1: Clock stretching disabled

Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 16 **SBC**: Target byte control

This bit is used to enable hardware byte control in target mode.
0: Target byte control disabled
1: Target byte control enabled

Bit 15 **RXDMAEN**: DMA reception requests enable

0: DMA mode disabled for reception
1: DMA mode enabled for reception

Bit 14 **TXDMAEN**: DMA transmission requests enable

0: DMA mode disabled for transmission
1: DMA mode enabled for transmission

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ANFOFF:** Analog noise filter OFF

0: Analog noise filter enabled
1: Analog noise filter disabled

Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bits 11:8 **DNF[3:0]**: Digital noise filter

These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter, filters spikes with a length of up to DNF[3:0] * t_{I2CCLK}

0000: Digital filter disabled

0001: Digital filter enabled and filtering capability up to one t_{I2CCLK}

...

1111: digital filter enabled and filtering capability up to fifteen t_{I2CCLK}

Note: If the analog filter is enabled, the digital filter is added to it. This filter can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 7 **ERRIE**: Error interrupts enable

0: Error detection interrupts disabled

1: Error detection interrupts enabled

Note: Any of these errors generates an interrupt:

- arbitration loss (ARLO)

- bus error detection (BERR)

- overrun/underrun (OVR)

- timeout detection (TIMEOUT)

- PEC error detection (PECERR)

- alert pin event detection (ALERT)

Bit 6 **TCIE**: Transfer complete interrupt enable

0: Transfer complete interrupt disabled

1: Transfer complete interrupt enabled

Note: Any of these events generates an interrupt:

Transfer complete (TC)

Transfer complete reload (TCR)

Bit 5 **STOPIE**: STOP detection interrupt enable

0: STOP detection (STOPF) interrupt disabled

1: STOP detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received interrupt enable

0: Not acknowledge (NACKF) received interrupts disabled

1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match interrupt enable (target only)

0: Address match (ADDR) interrupts disabled

1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX interrupt enable

0: Receive (RXNE) interrupt disabled

1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX interrupt enable

0: Transmit (TXIS) interrupt disabled

1: Transmit (TXIS) interrupt enabled

Bit 0 **PE**: Peripheral enable

0: Peripheral disabled

1: Peripheral enabled

Note: When PE = 0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least three APB clock cycles.

35.9.2 I2C control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]							
					rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **PECBYTE**: Packet error checking byte

This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE = 0.

0: No PEC transfer

1: PEC transmission/reception is requested

Note: Writing 0 to this bit has no effect.

This bit has no effect when RELOAD is set, and in target mode when SBC = 0.

Bit 25 **AUTOEND**: Automatic end mode (controller mode)

This bit is set and cleared by software.

0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.

1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

Note: This bit has no effect in target mode or when the RELOAD bit is set.

Bit 24 **RELOAD**: NBYTES reload mode

This bit is set and cleared by software.

0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).

1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in target mode with SBC = 0.

Note: Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (target mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE = 0.

0: an ACK is sent after current received byte.

1: a NACK is sent after current received byte.

Note: Writing 0 to this bit has no effect.

This bit is used only in target mode: in controller receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in target receiver NOSTRETCH mode, a NACK is automatically generated, whatever the NACK bit value.

When hardware PEC checking is enabled (PECBYTE = 1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: STOP condition generation

This bit only pertains to controller mode. It is set by software and cleared by hardware when a STOP condition is detected or when PE = 0.

0: No STOP generation

1: STOP generation after current byte transfer

Note: Writing 0 to this bit has no effect.

Bit 13 **START**: START condition generation

This bit is set by software. It is cleared by hardware after the START condition followed by the address sequence is sent, by an arbitration loss, by an address matched in target mode, by a timeout error detection, or when PE = 0.

0: No START generation

1: RESTART/START generation:

If the I2C is already in controller mode with AUTOEND = 0, setting this bit generates a repeated START condition when RELOAD = 0, after the end of the NBYTES transfer.

Otherwise, setting this bit generates a START condition once the bus is free.

Note: Writing 0 to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in target mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (controller receiver mode)

0: The controller sends the complete 10-bit target address read sequence: START + 2 bytes 10-bit address in write direction + RESTART + first seven bits of the 10-bit address in read direction.

1: The controller sends only the first seven bits of the 10-bit address, followed by read direction.

Note: Changing this bit when the START bit is set is not allowed.

Bit 11 **ADD10**: 10-bit addressing mode (controller mode)

0: The controller operates in 7-bit addressing mode

1: The controller operates in 10-bit addressing mode

Note: Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (controller mode)

0: Controller requests a write transfer

1: Controller requests a read transfer

Note: Changing this bit when the START bit is set is not allowed.

Bits 9:0 **SADD[9:0]**: Target address (controller mode)

Condition: In 7-bit addressing mode (ADD10 = 0):
SADD[7:1] must be written with the 7-bit target address to be sent. Bits SADD[9], SADD[8] and SADD[0] are don't care.

Condition: In 10-bit addressing mode (ADD10 = 1):
SADD[9:0] must be written with the 10-bit target address to be sent.

Note: Changing these bits when the START bit is set is not allowed.

35.9.3 I2C own address 1 register (I2C_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA1EN**: Own address 1 enable

0: Own address 1 disabled. The received target address OA1 is NACKed.
1: Own address 1 enabled. The received target address OA1 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **OA1MODE**: Own address 1 10-bit mode

0: Own address 1 is a 7-bit address.
1: Own address 1 is a 10-bit address.

Note: This bit can be written only when OA1EN = 0.

Bits 9:0 **OA1[9:0]**: Interface own target address

7-bit addressing mode: OA1[7:1] contains the 7-bit own target address. Bits OA1[9], OA1[8] and OA1[0] are don't care.

10-bit addressing mode: OA1[9:0] contains the 10-bit own target address.

Note: These bits can be written only when OA1EN = 0.

35.9.4 I2C own address 2 register (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA2EN**: Own address 2 enable
0: Own address 2 disabled. The received target address OA2 is NACKed.
1: Own address 2 enabled. The received target address OA2 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **OA2MSK[2:0]**: Own address 2 masks
000: No mask
001: OA2[1] is masked and don't care. Only OA2[7:2] are compared.
010: OA2[2:1] are masked and don't care. Only OA2[7:3] are compared.
011: OA2[3:1] are masked and don't care. Only OA2[7:4] are compared.
100: OA2[4:1] are masked and don't care. Only OA2[7:5] are compared.
101: OA2[5:1] are masked and don't care. Only OA2[7:6] are compared.
110: OA2[6:1] are masked and don't care. Only OA2[7] is compared.
111: OA2[7:1] are masked and don't care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.

Note: These bits can be written only when OA2EN = 0.
As soon as OA2MSK ≠ 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged, even if the comparison matches.

Bits 7:1 **OA2[7:1]**: Interface address
7-bit addressing mode: 7-bit address

Note: These bits can be written only when OA2EN = 0.

Bit 0 Reserved, must be kept at reset value.

35.9.5 I2C timing register (I2C_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL[3:0]				SDADEL[3:0]			
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH[7:0]								SCLL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale i2c_ker_ck to generate the clock period t_{PRESC} used for data setup and hold counters (refer to section *I2C timings*), and for SCL high and low level counters (refer to section *I2C controller initialization*).

t_{PRESC} = (PRESC + 1) x t_{I2CCLK}

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay t_{SCLDEL} = (SCLDEL + 1) x t_{PRESC} between SDA edge and SCL rising edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SCLDEL}.

Note: t_{SCLDEL} is used to generate $t_{SU:DAT}$ timing.

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay t_{SDADEL} between SCL falling edge and SDA edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SDADEL}.

t_{SDADEL}= SDADEL x t_{PRESC}

Note: SDADEL is used to generate $t_{HD:DAT}$ timing.

Bits 15:8 **SCLH[7:0]**: SCL high period (controller mode)

This field is used to generate the SCL high period in controller mode.

t_{SCLH} = (SCLH + 1) x t_{PRESC}

Note: SCLH is also used to generate $t_{SU:STO}$ and $t_{HD:STA}$ timing.

Bits 7:0 **SCLL[7:0]**: SCL low period (controller mode)

This field is used to generate the SCL low period in controller mode.

t_{SCLL} = (SCLL + 1) x t_{PRESC}

Note: SCLL is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

Note: This register must be configured when the I2C peripheral is disabled (PE = 0).

Note: The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

35.9.6 I2C timeout register (I2C_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TEXTEN**: Extended clock timeout enable

0: Extended clock timeout detection is disabled

1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than $t_{LOW:EXT}$ is done by the I2C interface, a timeout error is detected (TIMEOUT = 1).

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B

This field is used to configure the cumulative clock extension timeout:

– Controller mode: the controller cumulative clock low extend time ($t_{LOW:MEXT}$) is detected

– Target mode: the target cumulative clock low extend time ($t_{LOW:SEXT}$) is detected

$t_{LOW:EXT}$ = (TIMEOUTB + TIDLE = 01) x 2048 x t_{I2CCLK}

Note: These bits can be written only when TEXTEN = 0.

Bit 15 **TIMOUTEN**: Clock timeout enable

0: SCL timeout detection is disabled

1: SCL timeout detection is enabled. When SCL is low for more than $t_{TIMEOUT}$ (TIDLE = 0) or high for more than t_{IDLE} (TIDLE = 1), a timeout error is detected (TIMEOUT = 1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TIDLE**: Idle clock timeout detection

0: TIMEOUTA is used to detect SCL low timeout

1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)

Note: This bit can be written only when TIMOUTEN = 0.

Bits 11:0 **TIMEOUTA[11:0]**: Bus timeout A

This field is used to configure:

The SCL low timeout condition $t_{TIMEOUT}$ when TIDLE = 0

$t_{TIMEOUT}$= (TIMEOUTA + 1) x 2048 x t_{I2CCLK}

The bus idle condition (both SCL and SDA high) when TIDLE = 1

t_{IDLE}= (TIMEOUTA + 1) x 4 x t_{I2CCLK}

Note: These bits can be written only when TIMOUTEN = 0.

35.9.7 I2C interrupt and status register (I2C_ISR)

Address offset: 0x18

Reset value: 0x0000 0001

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r		r	r	r	r	r	r	r	r	r	r	r	r	rs	rs

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **ADDCODE[6:0]**: Address match code (target mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1). In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the two MSBs of the address.

Bit 16 **DIR**: Transfer direction (target mode)

This flag is updated when an address match event occurs (ADDR = 1).

0: Write transfer, target enters receiver mode.

1: Read transfer, target enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected, and cleared by hardware when a STOP condition is detected, or when PE = 0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN = 1 (SMBus host configuration), ALERTEN = 1 and an SMBALERT# event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 12 **TIMEOUT**: Timeout or t_{LOW} detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 11 **PECERR**: PEC error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 10 **OVR**: Overrun/underrun (target mode)

This flag is set by hardware in target mode with NOSTRETCH = 1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 9 **ARLO**: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 8 **BERR**: Bus error

This flag is set by hardware when a misplaced START or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in target mode. It is cleared by software by setting the BERRCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 7 **TCR**: Transfer complete reload

This flag is set by hardware when RELOAD = 1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

Note: This bit is cleared by hardware when PE = 0.

This flag is only for controller mode, or for target mode when the SBC bit is set.

Bit 6 **TC**: Transfer complete (controller mode)

This flag is set by hardware when RELOAD = 0, AUTOEND = 0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

Note: This bit is cleared by hardware when PE = 0.

Bit 5 **STOPF**: STOP detection flag

This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:

- as a controller, provided that the STOP condition is generated by the peripheral.
- as a target, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 4 **NACKF**: Not acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 3 **ADDR**: Address matched (target mode)

This bit is set by hardware as soon as the received target address matched with one of the enabled target addresses. It is cleared by software by setting *ADDRCF bit.*

Note: This bit is cleared by hardware when PE = 0.

Bit 2 **RXNE**: Receive data register not empty (receivers)

This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read.

Note: This bit is cleared by hardware when PE = 0.

Bit 1 **TXIS**: Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software only when NOSTRETCH = 1, to generate a TXIS event (interrupt if TXIE = 1 or DMA request if TXDMAEN = 1).

Note: This bit is cleared by hardware when PE = 0.

Bit 0 **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software in order to flush the transmit data register I2C_TXDR.

Note: This bit is set by hardware when PE = 0.

35.9.8 I2C interrupt clear register (I2C_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **ALERTCF**: Alert flag clear
Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.

Bit 12 **TIMOUTCF**: Timeout detection flag clear
Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.

Bit 11 **PECCF**: PEC error flag clear
Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.

Bit 10 **OVRCF**: Overrun/underrun flag clear
Writing 1 to this bit clears the OVR flag in the I2C_ISR register.

Bit 9 **ARLOCF**: Arbitration lost flag clear
Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.

Bit 8 **BERRCF**: Bus error flag clear
Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STOPCF**: STOP detection flag clear
Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.

Bit 4 **NACKCF**: Not acknowledge flag clear
Writing 1 to this bit clears the NACKF flag in I2C_ISR register.

Bit 3 **ADDRCF**: Address matched flag clear
Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.

Bits 2:0 Reserved, must be kept at reset value.

35.9.9 I2C PEC register (I2C_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: no wait states

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PEC[7:0]:** Packet error checking register
This field contains the internal PEC when PECEN=1.
The PEC is cleared by hardware when PE = 0.

35.9.10 I2C receive data register (I2C_RXDR)

Address offset: 0x24

Reset value: 0x0000 0000

Access: no wait states

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RXDATA[7:0]:** 8-bit receive data
Data byte received from the I²C-bus.

35.9.11 I2C transmit data register (I2C_TXDR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: no wait states

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXDATA[7:0]:** 8-bit transmit data

Data byte to be transmitted to the I²C-bus

Note: These bits can be written only when TXE = 1.

35.9.12 I2C autonomous mode control register (I2C_AUTOCR)

Address offset: 0x2C

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGEN	TRIGPOL	TRIGSEL[3:0]			
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCRDMAEN	TCDMAEN	Res.	Res.	Res.	Res.	Res.	Res.
								rw	rw						

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **TRIGEN:** Trigger enable

0: Trigger disabled

1: Trigger enabled

When a trigger is detected, a START condition is sent and the transfer is launched as defined in I2C_CR2.

Bit 20 **TRIGPOL:** Trigger polarity

0: Trigger active on rising edge

1: Trigger active on falling edge

Note: This bit can be written only when PE = 0

Bits 19:16 **TRIGSEL[3:0]:** Trigger selection (refer to *Section 35.4.2: I2C pins and internal signals* I2C interconnections tables).
0000: i2c_trg0 selected
0001: i2c_trg1 selected
...
1111: i2c_trg15 selected
Note: This bit can be written only when PE = 0

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TCRDMAEN**: DMA request enable on Transfer Complete Reload event
0: DMA request not generated on Transfer Complete Reload event
1: DMA request generated on Transfer Complete Reload event

Bit 6 **TCDMAEN**: DMA request enable on Transfer Complete event
0: DMA request not generated on Transfer Complete event
1: DMA request generated on Transfer Complete event

Bits 5:0 Reserved, must be kept at reset value.

35.9.13 I2C register map

The table below provides the I2C register map and the reset values.

Table 341. I2C register map and reset values

Offset	Register name	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
0x00	I2C_CR1	STOPFACLR	ADDRACLR	Res.	Res.	Res.	Res.	Res.	FMP	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
	Reset value	0	0						0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	I2C_CR2	Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	I2C_OAR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0
0x0C	I2C_OAR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA2EN	Res.	Res.	Res.	Res.	OA2MSK [2:0]			OA2[7:1]							Res.
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	
0x10	I2C_TIMINGR	PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL [3:0]				SDADEL [3:0]				SCLH[7:0]								SCLL[7:0]							
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	I2C_TIMEOUTR	TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]												TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	I2C_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
	Reset value									0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	I2C_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
	Reset value																			0	0	0	0	0	0			0	0	0			
0x20	I2C_PECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x24	I2C_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	I2C_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 341. I2C register map and reset values (continued)

Offset	Register name	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
0x2C	I2C_AUTOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGEN	TRIGPOL	TRIGSEL [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCRDMAEN	TCDMAEN	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0	0	0	0	0	0									0	0						

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

36 Universal synchronous/asynchronous receiver transmitter (USART/UART)

36.1 USART introduction

The USART offers a flexible means to perform full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and half-duplex single-wire communications, as well as LIN (local interconnection network), smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.

36.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data

  Each FIFO can be enabled/disabled by software and come with a status flag.
- A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous master/slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Communication control/error detection flags
- Parity control:
  – Transmits parity bit

  - Checks parity of received data byte
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode
- Autonomous functionality in Stop mode

36.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T = 0 and T = 1 asynchronous protocols for smartcards as defined in the ISO/IEC 7816-3 standard
  - 0.5 and 1.5 stop bits for smartcard operation
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

36.4 USART implementation

The following tables describe the USART implementation. LPUART is included for comparison.

Table 342. Instance implementation on STM32WBA2

Instance	STM32WBA2
USART1	Full
LPUART1	Low-power

Table 343. USART/LPUART features

Modes/features(1)	Full feature set	Basic feature set	Low-power feature set
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (master/slave)	X	-	-
Smartcard mode	X	-	-
Single-wire half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-

Table 343. USART/LPUART features (continued)

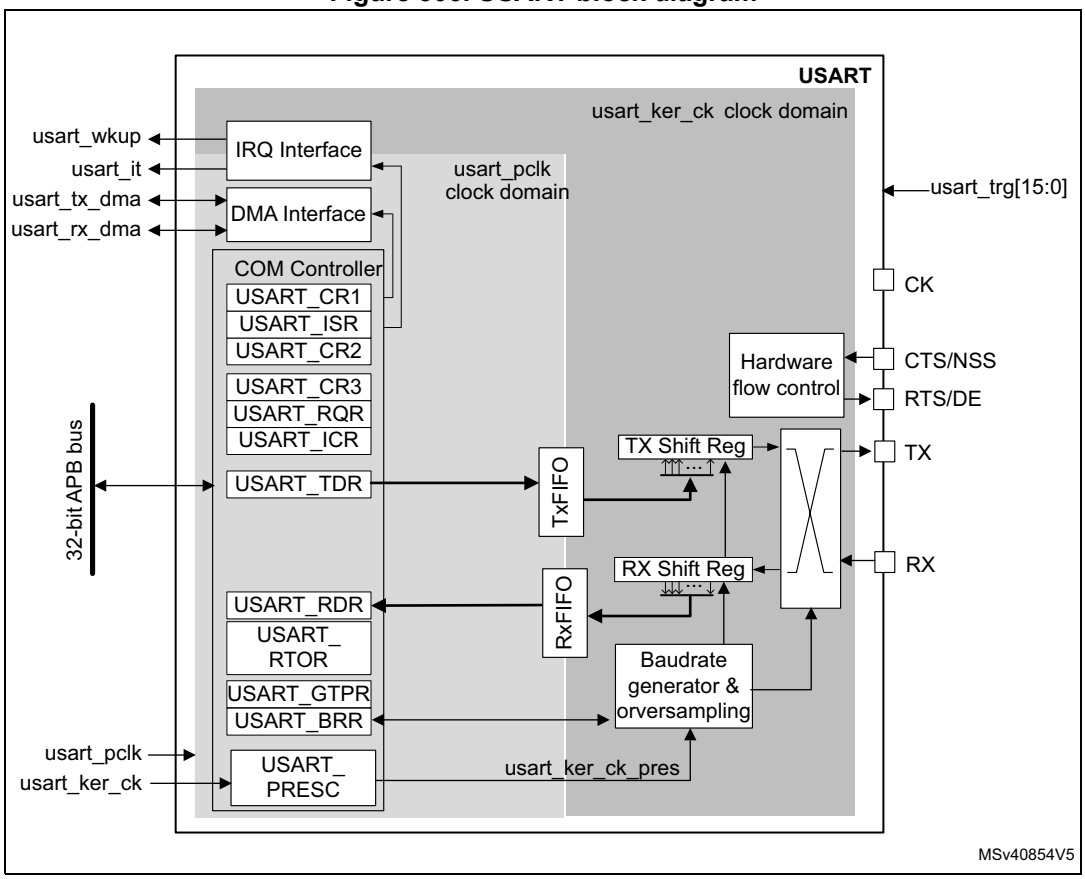
Modes/features[1]	Full feature set	Basic feature set	Low-power feature set
LIN mode	X	X	-
Dual clock domain	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	8		
Wake-up from low-power mode	X[2]	X[2]	X[3]
Autonomous mode	X	X	X

1. “X” = supported, “-” = not supported.
2. Wake-up supported from Stop 0 and Stop 1 modes.
3. Wake-up supported from Stop 0, Stop 1, and Stop 2 modes.

36.5 USART functional description

36.5.1 USART block diagram

Figure 306. USART block diagram



36.5.2 USART pins and internal signals

Description of USART input/output pins

- USART bidirectional communications

  USART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

  – **RX** (Receive Data Input)

    RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate between valid incoming data and noise.

  – **TX** (Transmit Data Output)

    When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High. In single-wire and smartcard modes, this I/O is used to transmit and receive data.

- RS232 hardware flow control mode

  The following pins are required in RS232 hardware flow control mode:

  – **CTS** (Clear To Send)

    When driven high, this signal blocks the data transmission at the end of the current transfer.

  – **RTS** (Request To Send)

    When it is low, this signal indicates that the USART is ready to receive data.

- RS485 hardware control mode

  The **DE** (Driver Enable) pin is required in RS485 hardware control mode. This signal activates the transmission mode of the external transceiver.

- Synchronous SPI master/slave mode and smartcard mode

  The following pins are required in synchronous master/slave mode and smartcard mode:

  – **CK**

    This pin acts as clock output in synchronous SPI master and smartcard modes.

    It acts as clock input in synchronous SPI slave mode.

    In synchronous master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX pin. This mechanism can be used to control peripherals featuring shift registers (such as LCD drivers). The clock phase and polarity are software programmable.

    In smartcard mode, CK output provides the clock to the smartcard.

  – **NSS**

    This pin acts as Slave Select input in synchronous slave mode.

Refer to *Table 344* and *Table 345* for the list of USART input/output pins and internal signals.

Table 344. USART/UART input/output pins

Pin name	Signal type	Description
USART_RX/UART_RX	Input	Serial data receive input
USART_TX/UART_TX	Output	Transmit data output
USART_CTS/UART_CTS	Input	Clear to send
USART_RTS/UART_RTS	Output	Request to send
USART_DE(1)/UART_DE(2)	Output	Driver enable
USART_CK	Output	Clock output in synchronous master and smartcard modes.
USART_NSS(3)	Input	Slave select input in synchronous slave mode.

1. USART_DE and USART_RTS share the same pin.
2. UART_DE and UART_RTS share the same pin.
3. USART_NSS and USART_CTS share the same pin.

Description of USART input/output signals

Table 345. USART internal input/output signals

Signal name	Signal type	Description
usart_pclk	Input	APB clock
usart_ker_ck	Input	USART kernel clock
usart_wkup	Output	USART provides a wake-up interrupt
usart_it	Output	USART global interrupt
usart_tx_dma	Input/output	USART transmit DMA request
usart_rx_dma	Input/output	USART receive DMA request
usart_trg[15:0]	Input	USART triggers.

Description of USART interconnections

Table 346. USART interconnection (USART1)

Signal name	Source
usart_trg0	lpdma1_ch0_tc
usart_trg1	lpdma1_ch1_tc
usart_trg2	lpdma1_ch2_tc
usart_trg3	lpdma1_ch3_tc
usart_trg4	exti6
usart_trg5	exti9
usart_trg6	lptim1_ch1
usart_trg7	lptim2_ch1
usart_trg8	-
usart_trg9	-
usart_trg10	rtc_alra_trg
usart_trg11	rtc_wut_trg
usart_trg12	-
usart_trg13	-
usart_trg14	-
usart_trg15	-

36.5.3 USART clocks

The simplified block diagram given in *Section 36.5.1: USART block diagram* shows two fully-independent clock domains:

- The **usart_pclk** clock domain

  The **usart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.
- The **usart_ker_ck** kernel clock domain.

  The **usart_ker_ck** is the USART clock source. It is independent from **usart_pclk** and delivered by the RCC. The USART registers can consequently be written/read even when the **usart_ker_ck** clock is stopped.

  When the dual clock domain feature is not supported, the **usart_ker_ck** clock is the same as the **usart_pclk** clock.

There is no constraint between **usart_pclk** and **usart_ker_ck**: **usart_ker_ck** can be faster or slower than **usart_pclk**. The only limitation is the software ability to manage the communication fast enough.

When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external CK signal provided by the external master SPI device. The **usart_ker_ck** clock must be at least 3 times faster than the clock on the CK input.

36.5.4 USART character description

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the USART_CR1 register (see *Figure 307*):

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

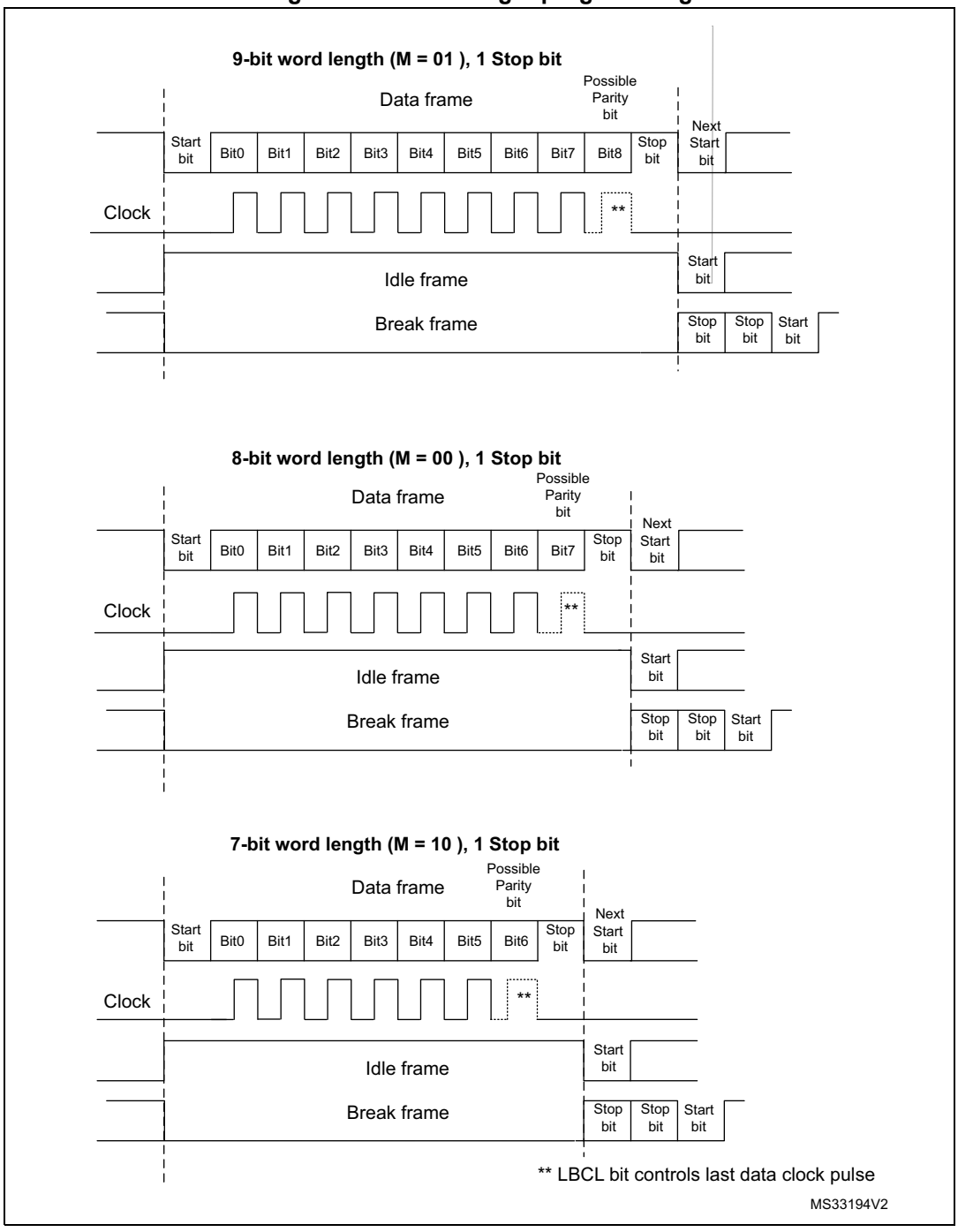
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clock are generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.

Figure 307. Word length programming



36.5.5 USART FIFOs and thresholds

The USART can operate in FIFO mode.

The USART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in USART_CR1 register (bit 29). This mode is supported only in UART, SPI and smartcard modes.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the USART_ISR register.

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through the RXFTCFG[2:0] and TXFTCFG[2:0] bitfields of the USART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield.

  In this case, the RXFT flag is set in the USART_ISR register. This means that RXFTCFG[2:0] data have been received: 1 data in USART_RDR and (RXFTCFG[2:0] – 1) data in the RXFIFO. As an example, when RXFTCFG[2:0] is programmed to 101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received (FIFO size – 1 data in the RXFIFO and 1 data in the USART_RDR). As a result, the next received data does not set the overrun flag.
- The Tx interrupt is generated when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of the USART_CR3 register.

36.5.6 USART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin while the corresponding clock pulses are output on the CK pin.

Character transmission

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART_TDR) are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the USART_TDR.

The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost.

An idle frame is sent when the TE bit is enabled.

Configurable stop bits

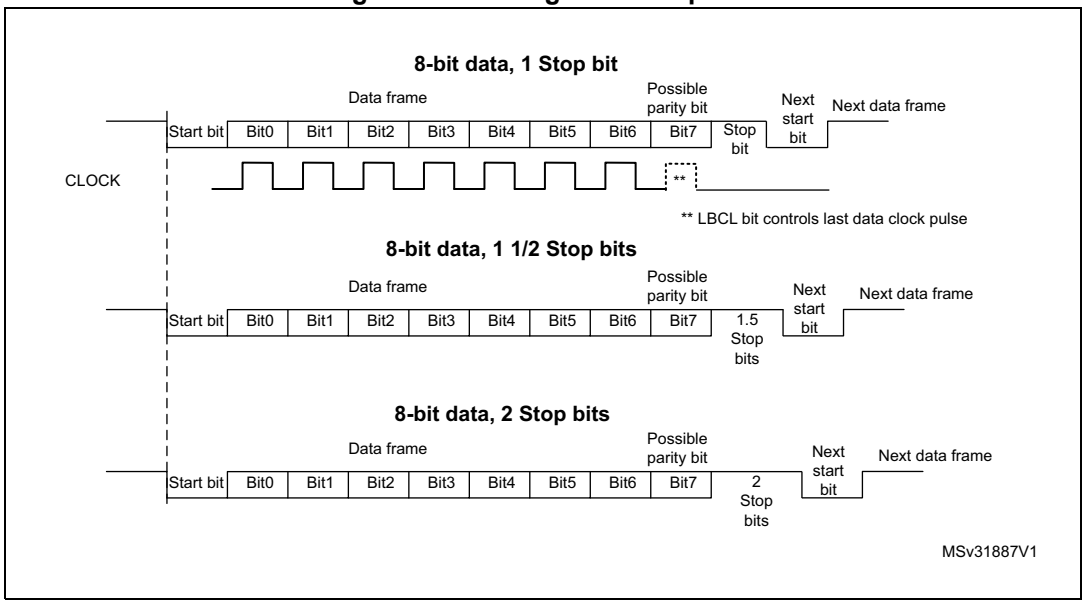
The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.

- ***1 stop bit***: This is the default value of number of stop bits.
- ***2 stop bits***: This is supported by normal USART, single-wire and modem modes.
- ***1.5 stop bits***: To be used in smartcard mode.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits (see *Section 36.5.1: USART block diagram*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 308. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the USART_BRR register.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAT) in USART_CR3 if multibuffer communication must take place. Configure the DMA register as explained in *Section 36.5.20: Continuous communication using USART and DMA*.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.

7. Write the data to send in the USART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data to the USART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data to the USART_TDR adds one data to the TXFIFO. Write operations to the USART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the USART_TDR register, wait until TC = 1.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame has completed.
   – When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the USART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode is disabled

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:
  – the data have been moved from the USART_TDR register to the shift register and the data transmission has started;
  – the USART_TDR register is empty;
  – the next data can be written to the USART_TDR register without overwriting the previous data.

  This flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the USART_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the USART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.
- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  – the TXFIFO is not full;
  – the USART_TDR register is empty;
  – the next data can be written to the USART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the USART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

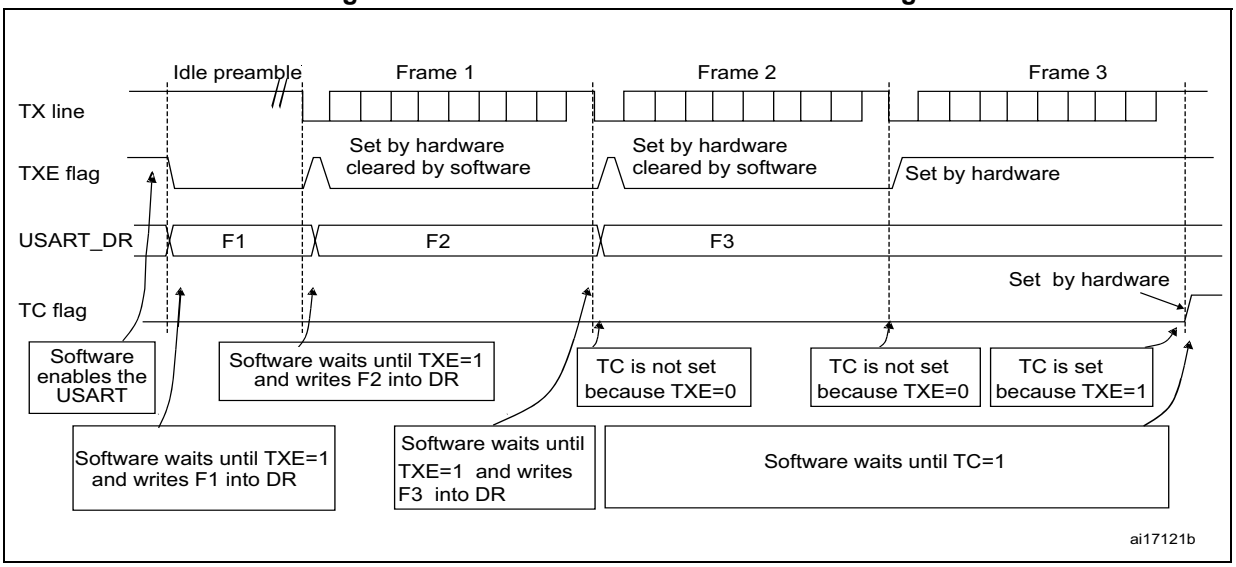
  When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

  Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger level.

  If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

After writing the last data to the USART_TDR register, it is mandatory to wait until TC is set before disabling the USART or causing the microcontroller to enter the low-power mode (see *Figure 309*).

Figure 309. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see *Figure 307*).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is complete (during the stop bits after the break character). The USART inserts a logic 1 signal (stop) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

36.5.7 USART receiver

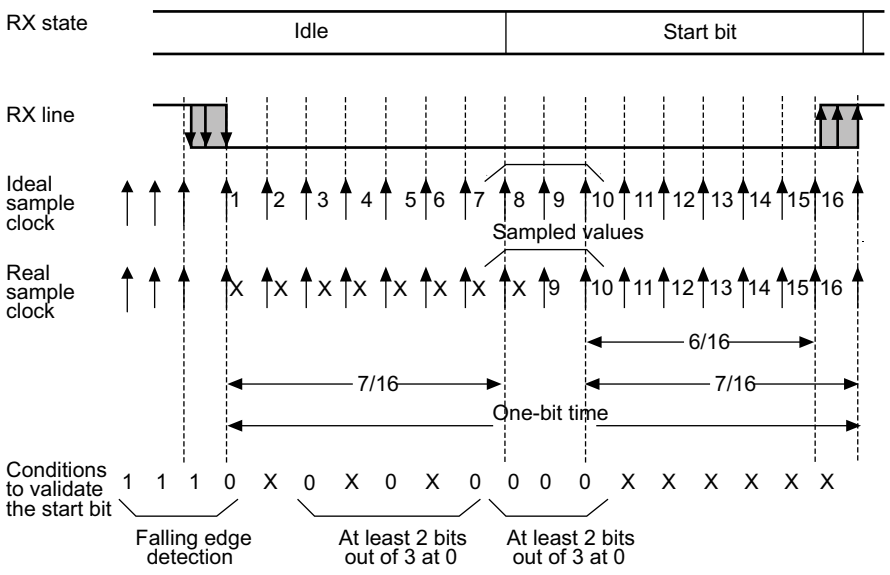
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0.

Figure 310. Start bit detection when oversampling by 16 or 8



Note: If the sequence has not completed, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set and interrupt generated if RXNEIE = 1, or RXFNE flag set and interrupt generated if RXFNEIE = 1 if FIFO mode enabled) if the 3 sampled bits are at 0 (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at 0 and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at 0).

The start bit is validated but the NE noise flag is set if,

- for both samplings, 2 out of the 3 sampled bits are at 0 (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits), or
- for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at 0.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

Character reception

During an USART reception, data are shifted out least significant bit first (default configuration) through the RX pin.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART_BRR
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 36.5.20: Continuous communication using USART and DMA*.
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the USART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads the Receive data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. A DMA request is triggered when the RXFIFO is not empty, that is when there are data to be read from the RXFIFO.
- In single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the USART_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to 1 in the USART_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read operation from USART_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by programming RXFRQ bit to 1 in USART_RQR. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character, to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be

generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs if a character is received and RXNE has not been reset.

  Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXN E flag is set after every byte reception.

  An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – the ORE bit is set;
  – the RDR content is not lost. The previous data is available by reading the USART_RDR register.
  – the shift register is overwritten. After that, any data received during overrun is lost.
  – an interrupt is generated if either the RXNEIE or the EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full.

  Data can not be transferred from the shift register to the USART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set.
  – The first entry in the RXFIFO is not lost. It is available by reading the USART_RDR register.
  – The shift register is overwritten. After that point, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the USART_ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost.

When the FIFO mode is disabled, there are two possibilities

- *if RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE = 0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.*

Selecting the clock source and the appropriate oversampling method

The choice of the clock source is done through the Clock Control system (see *Section : Reset and Clock Control (RCC)*). The clock source must be selected through the UE bit before enabling the USART.

The clock source must be selected according to two criteria:

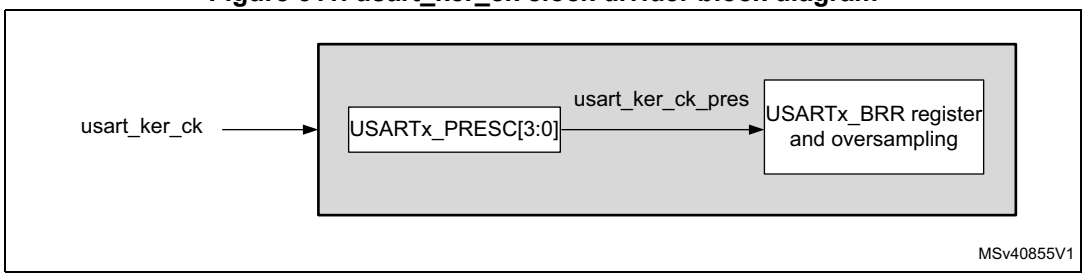
- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is usart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the usart_ker_ck clock source can be configurable in the RCC (see *Section : Reset and Clock Control (RCC)*). Otherwise the usart_ker_ck clock is the same as usart_pclk.

The usart_ker_ck clock can be divided by a programmable factor, defined in the USART_PRESC register.

Figure 311. usart_ker_ck clock divider block diagram



Some usart_ker_ck sources enable the USART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selected, the USART wakes up the MCU, when needed, in order to transfer the received data, by performing a software read to the USART_RDR register or by DMA.

For the other clock sources, the system must be active to enable USART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This enables obtaining the best a trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register either to 16 or 8 times the baud rate clock (see *Figure 312* and *Figure 313*).

Depending on the application:

- select oversampling by 8 (OVER8 = 1) to achieve higher speed (up to usart_ker_ck_pres/8). In this case the maximum receiver tolerance to clock deviation is reduced (refer to *Section 36.5.9: Tolerance of the USART receiver to clock deviation*)
- select oversampling by 16 (OVER8 = 0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum usart_ker_ck_pres/16 (where usart_ker_ck_pres is the USART input clock divided by a prescaler).

Programming the ONEBIT bit in the USART_CR3 register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NE bit is set.
- A single sample in the center of the received bit

  Depending on the application:

  – select the three sample majority vote method (ONEBIT = 0) when operating in a noisy environment and reject the data when a noise is detected (refer to *Table 347*) because this indicates that a glitch occurred during the sampling.

  – select the single sample method (ONEBIT = 1) when the line is noise-free to increase the receiver tolerance to clock deviations (see *Section 36.5.9: Tolerance of the USART receiver to clock deviation*). In this case the NE bit is never set.

When noise is detected in a frame:

- The NE bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the USART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The NE bit is reset by setting NFCF bit in ICR register.

Note: Noise error is not supported in SPI and IrDA modes.

Oversampling by 8 is not available in the smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to 0 by hardware.

Figure 312. Data sampling when oversampling by 16

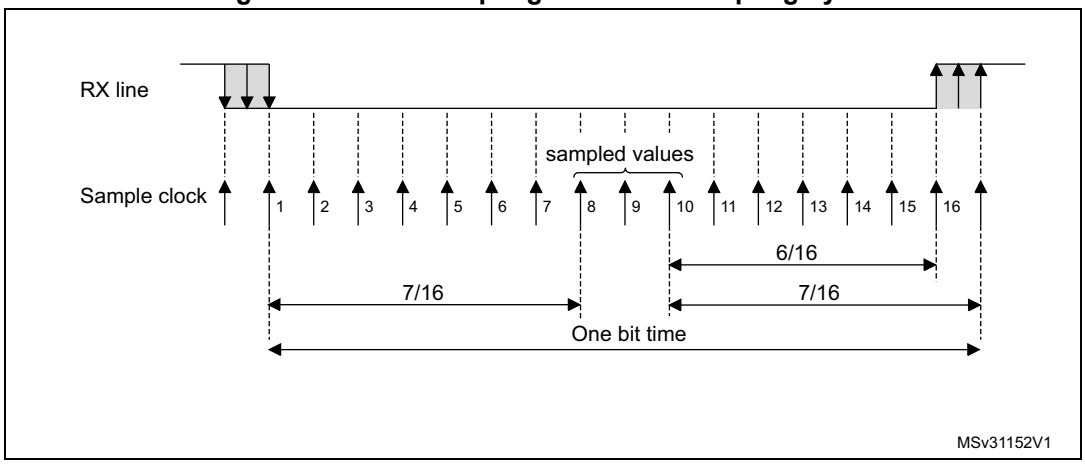


Figure 313. Data sampling when oversampling by 8

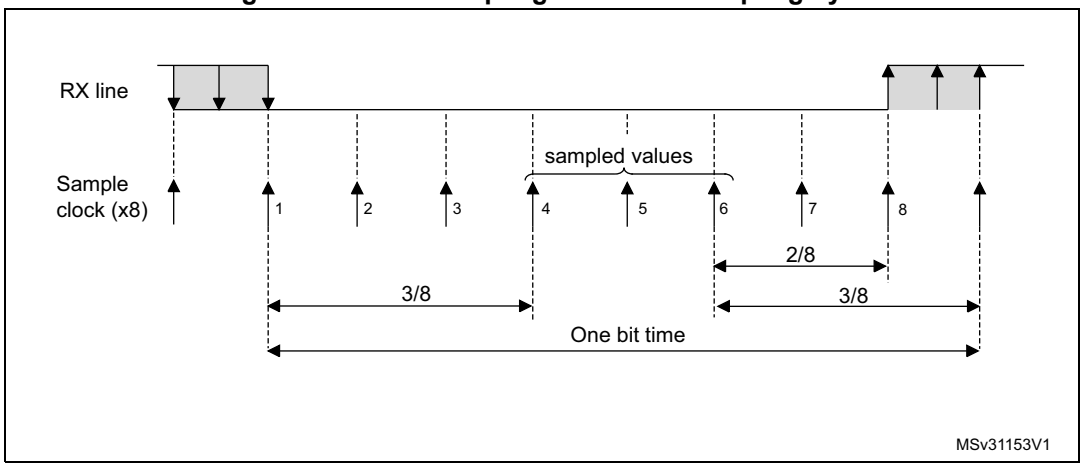


Table 347. Noise detection from sampled data

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware.
- the invalid data is transferred from the Shift register to the USART_RDR register (RXFIFO in case FIFO mode is enabled).
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the USART_ICR register.

Note: Framing error is not supported in SPI mode.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of USART_CR: it can be either 1 or 2 in normal mode and 0.5 or 1.5 in smartcard mode.

- **0.5 stop bit (reception in smartcard mode)**: no sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.
- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **1.5 stop bits (smartcard mode)**

  When transmitting in smartcard mode, the device must check that the data are correctly sent. The receiver block must consequently be enabled (RE = 1 in USART_CR1) and the stop bit is checked to test if the smartcard has detected a parity error.

  In the event of a parity error, the smartcard forces the data signal low during the sampling (NACK signal), which is flagged as a framing error. The FE flag is then set through RXNE flag (RXFNE if the FIFO mode is enabled) at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be broken into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through (refer to *Section 36.5.17: USART receiver timeout* for more details).
- **2 stop bits**

  Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit.

  The framing error flag is set if a framing error is detected during the first stop bit.

  The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

36.5.8 USART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the USART_BRR register.

Equation 1: baud rate for standard USART (SPI mode included) (OVER8 = 0 or 1)

In case of oversampling by 16, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

In case of oversampling by 8, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{2 \times \text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

Equation 2: baud rate in smartcard, LIN and IrDA modes (OVER8 = 0)

The baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1
  - BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
  - BRR[3] must be kept cleared.
  - BRR[15:4] = USARTDIV[15:4]

Note: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value must not be changed during communication.

In case of oversampling by 16 and 8, USARTDIV must be greater than or equal to 16.

How to derive USARTDIV from USART_BRR register values

Example 1

To obtain 9600 bauds with usart_ker_ck_pres= 8 MHz:

- In case of oversampling by 16:
  USARTDIV = 8 000 000/9600
  BRR[3:0] = USARTDIV = 0d833 = 0x0341
- In case of oversampling by 8:
  USARTDIV = 2 * 8 000 000/9600
  USARTDIV = 1666,66 (0d1667 = 0x683)
  BRR[3:0] = 0x3 >>1 = 0x1
  BRR[3:0] = 0x681

Example 2

To obtain 921.6 kbauds with usart_ker_ck_pres = 48 MHz:

- In case of oversampling by 16:
  USARTDIV = 48 000 000/921 600
  BRR[3:0] = USARTDIV = 52 = 0x34
- In case of oversampling by 8:
  USARTDIV = 2 * 48 000 000/921 600
  USARTDIV = 104 (0d104 = 0x68)
  BRR[3:0] = USARTDIV[3:0] >> 1 = 0x8 >> 1 = 0x4
  BRR[3:0] = 0x64

36.5.9 Tolerance of the USART receiver to clock deviation

The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.

The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < USART\ receiver\ tolerance$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WUUSART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WUUSART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WUUSART}}{9 \times Tbit}$$

$t_{WUUSART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 348*, *Table 349*, depending on the following settings:

- 9-, 10- or 11-bit character length defined by the M bits in the USART_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- Bits BRR[3:0] of USART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_CR3 register.

Table 348. Tolerance of the USART receiver when BRR [3:0] = 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.75%	4.375%	2.50%	3.75%
01	3.41%	3.97%	2.27%	3.41%
10	4.16%	4.86%	2.77%	4.16%

Table 349. Tolerance of the USART receiver when BRR[3:0] is different from 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.33%	3.88%	2%	3%
01	3.03%	3.53%	1.82%	2.73%
10	3.7%	4.31%	2.22%	3.33%

Note: The data specified in Table 348 and Table 349 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M= 01 or 9- bit times when M = 10).

36.5.10 USART auto baud rate detection

The USART can detect and automatically set the USART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system is using a relatively low accuracy clock source and this mechanism enables the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from usart_ker_ck_pres/65535 and usart_ker_ck_pres/16.
- When oversampling by 8, the baud rate ranges from usart_ker_ck_pres/32763 and usart_ker_ck_pres/8.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are the following:

- **Mode 0**: Any character starting with a bit at 1.

  In this case the USART measures the duration of the start bit (falling edge to rising edge).
- **Mode 1:** Any character starting with a 10xx bit pattern.

  In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.
- **Mode 2**: A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.
- **Mode 3**: A 0x55 character frame.

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a 0).

When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection must be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART_ISR register.

Note: The BRR value might be corrupted if the USART is disabled (UE = 0) during an auto baud rate operation.

36.5.11 USART multiprocessor communication

It is possible to perform USART multiprocessor communications (with several USARTs connected in a network). For instance one of the USARTs can be the master with its TX output connected to the RX inputs of the other USARTs, while the others are slaves with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non-addressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the USART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two usart_ker_ck cycles), otherwise mute mode might remain active.

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in USART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART_RQR register, under certain conditions.

The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:

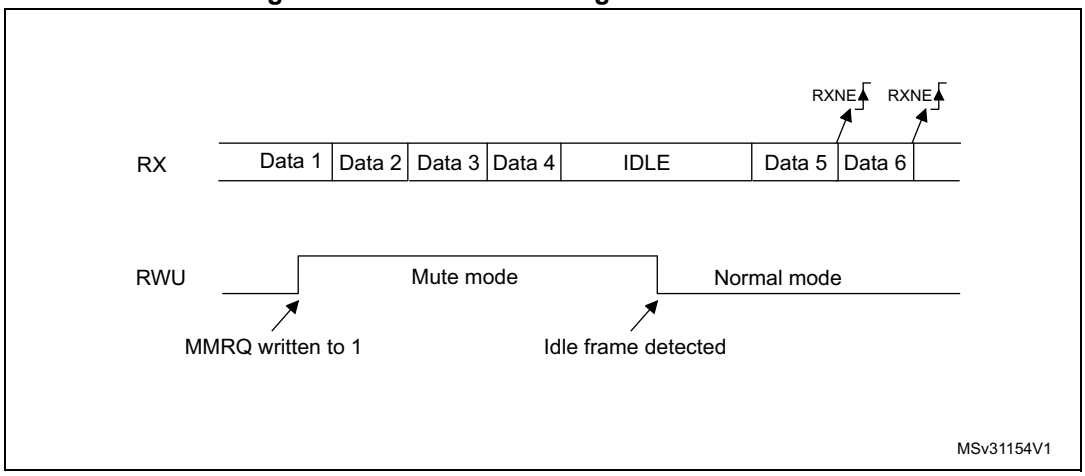
- Idle line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE = 0)

The USART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 314*.

Figure 314. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the USART is activated while the line is idle, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a 1, otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters mute mode. When FIFO management is enabled, the software must ensure that there is at least one empty location in the RXFIFO before entering mute mode.

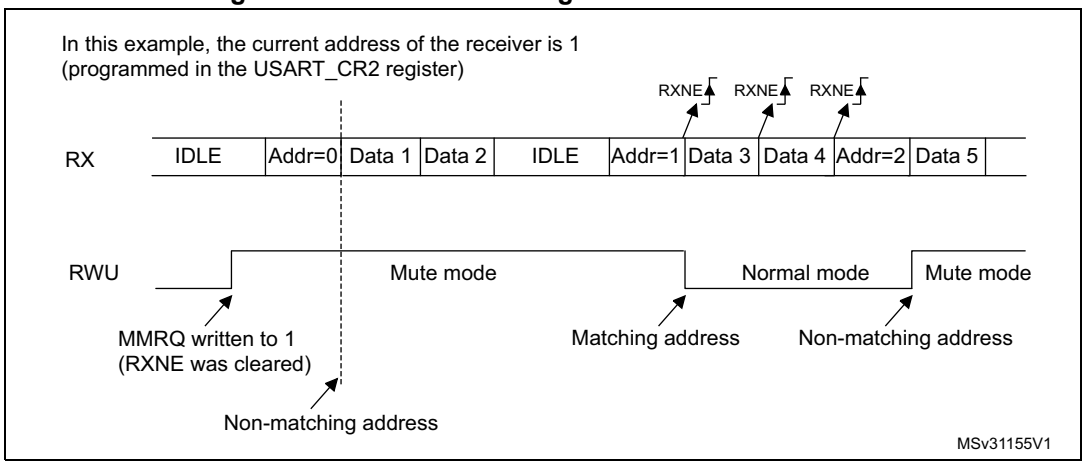
The USART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in mute mode

An example of mute mode behavior using address mark detection is given in *Figure 315*.

Figure 315. Mute mode using address mark detection



36.5.12 USART Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART_CR2 register and the RTOIE in the USART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception has not completed.

Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE = 1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

36.5.13 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in *Table 350*.

Table 350. USART frame formats

M bits	PCE bit	USART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1)).

36.5.14 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to *Section 36.4: USART implementation*.

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:

- STOP[1:0] and CLKEN in the USART_CR2 register,
- SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure described in *Section 36.5.5* has to be applied for LIN master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 zero bits as a break character. Then 2 bits of value ‘1 are sent to enable the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE = 1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL = 1 in USART_CR2) consecutive bits are detected as 0, and are followed by a delimiter character, the LBDF flag is set in USART_ISR. If the LBDIE bit = 1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a 1 is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN = 0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN = 1), as soon as a framing error occurs (that is, stop bit detected at 0, which is the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown in *Figure 316*.

Examples of break frames are given in *Figure 317*.

Figure 316. Break detection in LIN mode (11-bit break length - LBDL bit is set)

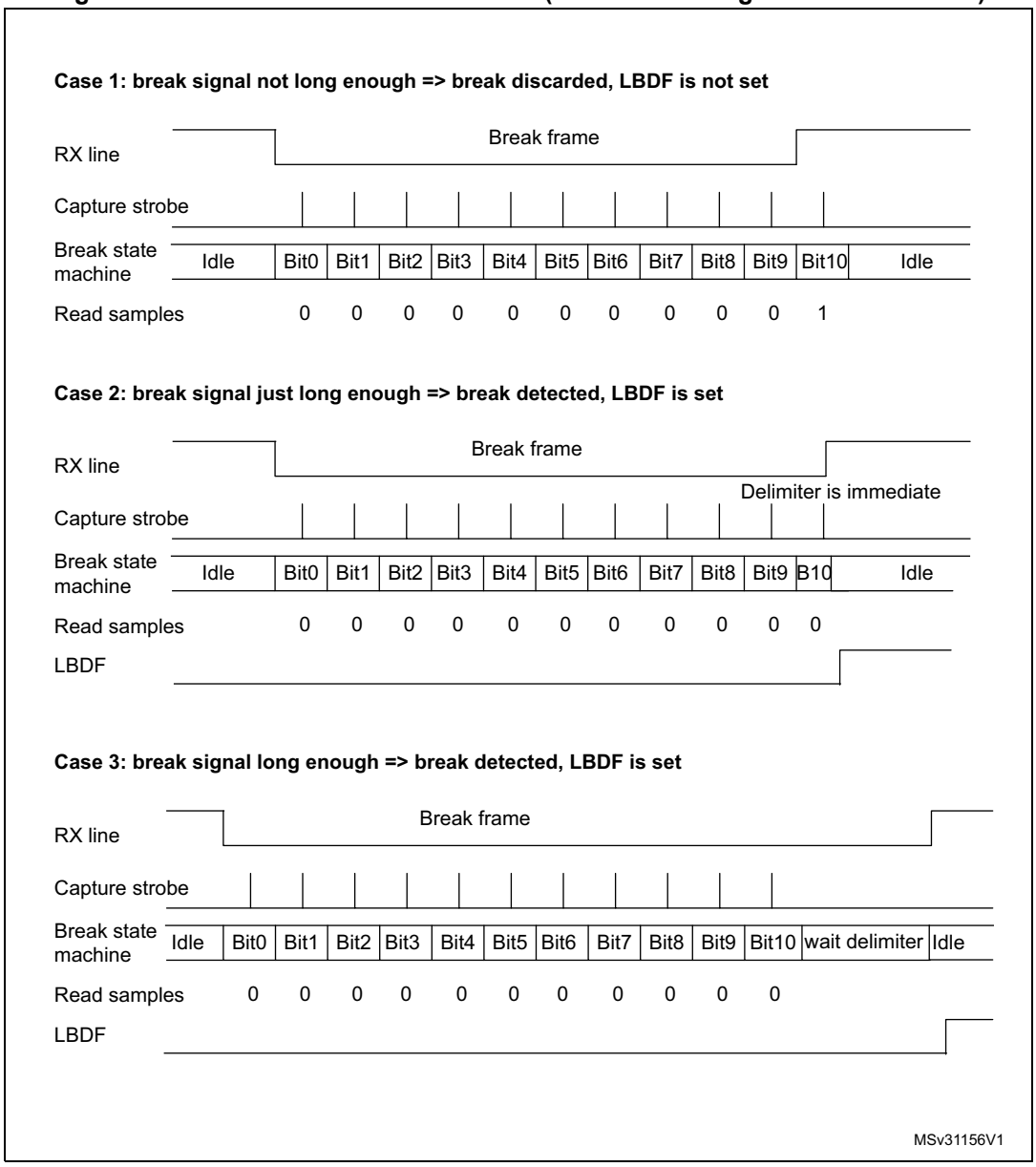
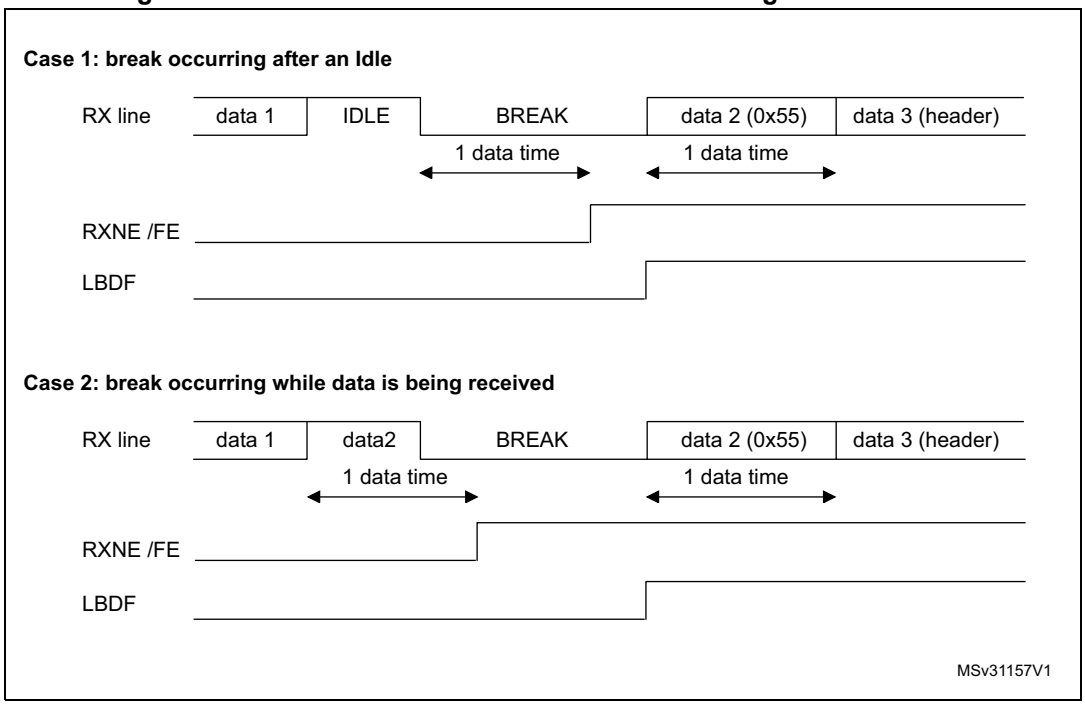


Figure 317. Break detection in LIN mode vs. Framing error detection



36.5.15 USART synchronous mode

Master mode

The synchronous master mode is selected by programming the CLKEN bit in the USART_CR2 register to 1. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see *Figure 318*, *Figure 319* and *Figure 320*).

During the Idle state, preamble and send break, the external CK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

Note: In master mode, the CK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE = 1) and data are being transmitted (USART_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.

Figure 318. USART example of synchronous master transmission

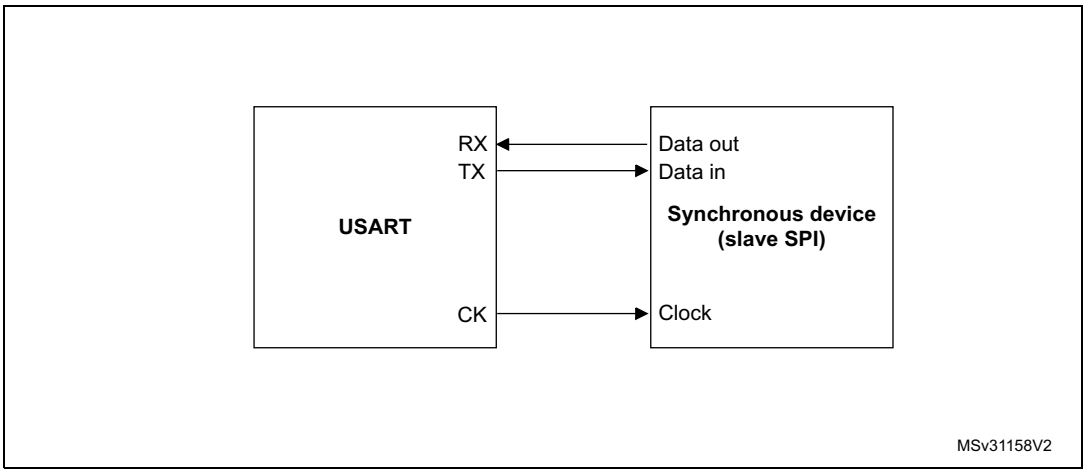


Figure 319. USART data clock timing diagram in synchronous master mode (M bits = 00)

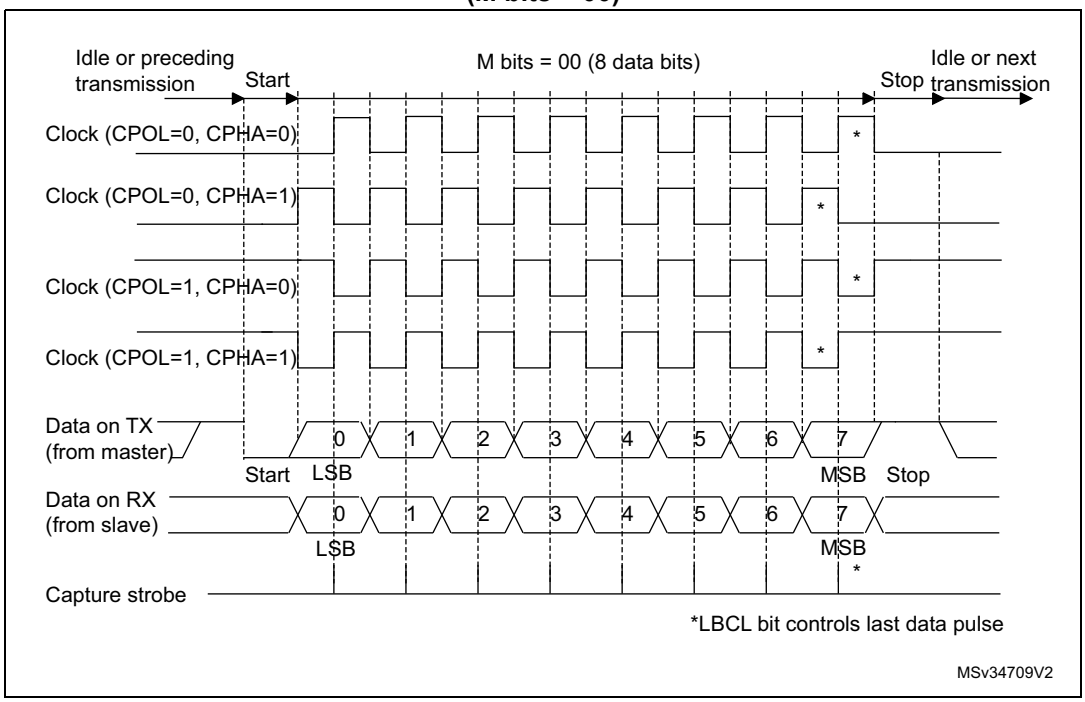
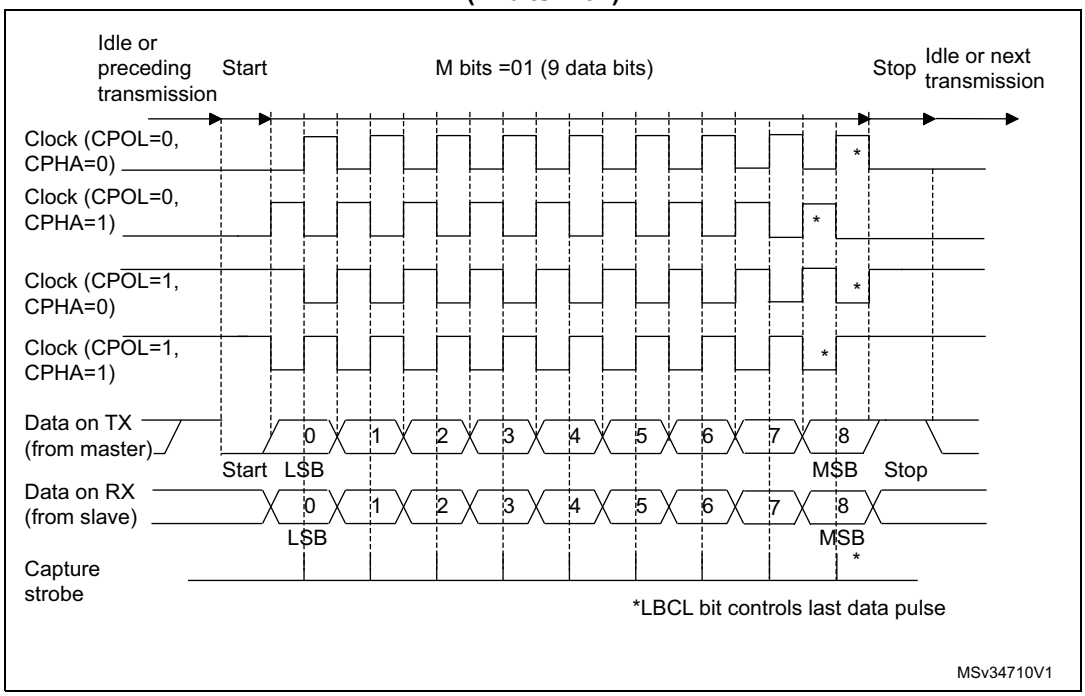


Figure 320. USART data clock timing diagram in synchronous master mode (M bits = 01)



Slave mode

The synchronous slave mode is selected by programming the SLVEN bit in the USART_CR2 register to 1. In synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in slave mode. The CK pin is the input of the USART in slave mode.

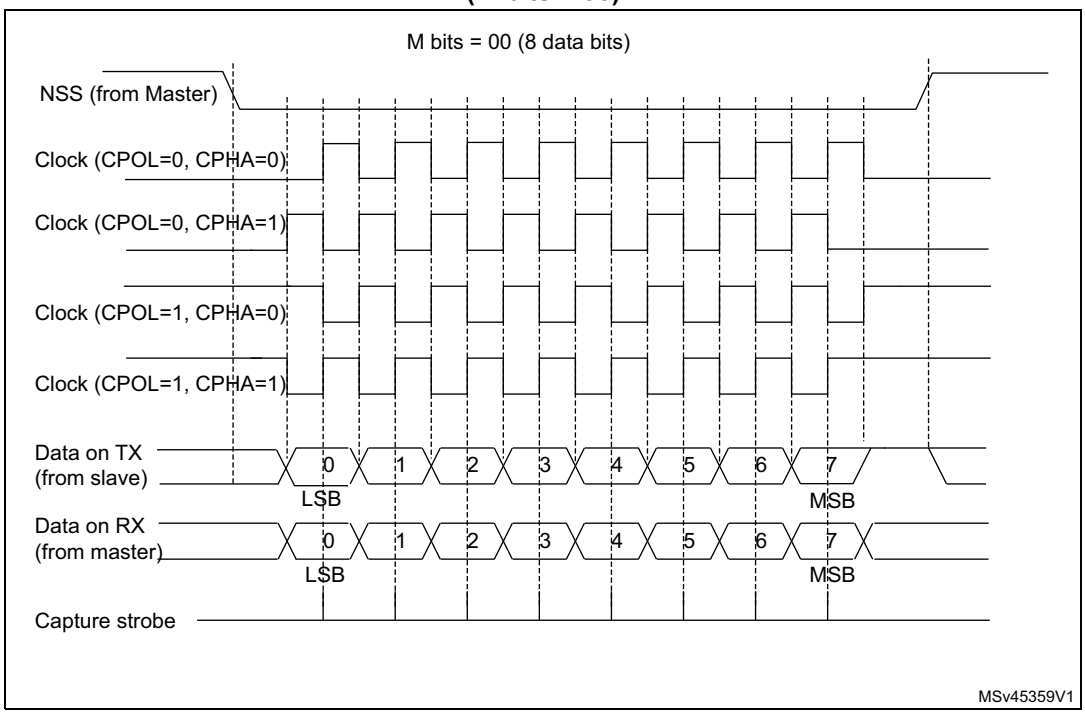
*Note: When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (*usart_ker_ck_pres*) must be greater than 3 times the CK input frequency.*

The CPOL bit and the CPHA bit in the USART_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see *Figure 321*).

An underrun error flag is available in slave transmission mode. This flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value to USART_TDR.

The slave supports the hardware and software NSS management.

Figure 321. USART data clock timing diagram in synchronous slave mode (M bits = 00)



Slave Select (NSS) pin management

The hardware or software slave select management can be set through the DIS_NSS bit in the USART_CR2 register:

- Software NSS management (DIS_NSS = 1)

  SPI slave is always selected and NSS input pin is ignored.

  The external NSS pin remains free for other application uses.
- Hardware NSS management (DIS_NSS = 0)

  The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS is high.

Note: The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE = 0) to ensure that the clock pulses function correctly.

In SPI slave mode, the USART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it becomes desynchronized with the master. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave transmits zeros.

SPI slave underrun error

When an underrun error occurs, the UDR flag is set in the USART_ISR register, and the SPI slave goes on sending the last data until the underrrun error flag is cleared by software.

The underrun flag is set at the beginning of the frame. An underrun error interrupt is triggered if EIE bit is set in the USART_CR3 register.

The underrun error flag is cleared by setting bit UDRCF in the USART_ICR register.

In case of underrun error, it is still possible to write to the TDR register. Clearing the underrun error enables sending new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

Note: An underrun error may occur if the moment the data is written to the USART_TDR is too close to the first CK transmission edge. To avoid this underrun error, the USART_TDR must be written 3 usart_ker_ck *cycles before the first CK edge.*

36.5.16 USART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in the USART_CR3 register.

The USART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in USART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

36.5.17 USART receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART_CR2 control register.

The timeout duration is programmed using the RTO bitfields in the USART_RTOR register.

The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = 00 or STOP = 11
- from the end of the second stop bit if STOP = 10.
- from the beginning of the stop bit if STOP = 01.

When the timeout duration has elapsed, the RTOF flag in the USART_ISR register is set. A timeout is generated if RTOIE bit in USART_CR1 register is set.

36.5.18 USART smartcard mode

This section is relevant only when smartcard mode is supported. Refer to *Section 36.4: USART implementation*.

Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

The CLKEN bit can also be set to provide a clock to the smartcard.

The smartcard interface is designed to support asynchronous smartcard protocol as defined in the ISO 7816-3 standard. Both T = 0 (character mode) and T = 1 (block mode) are supported.

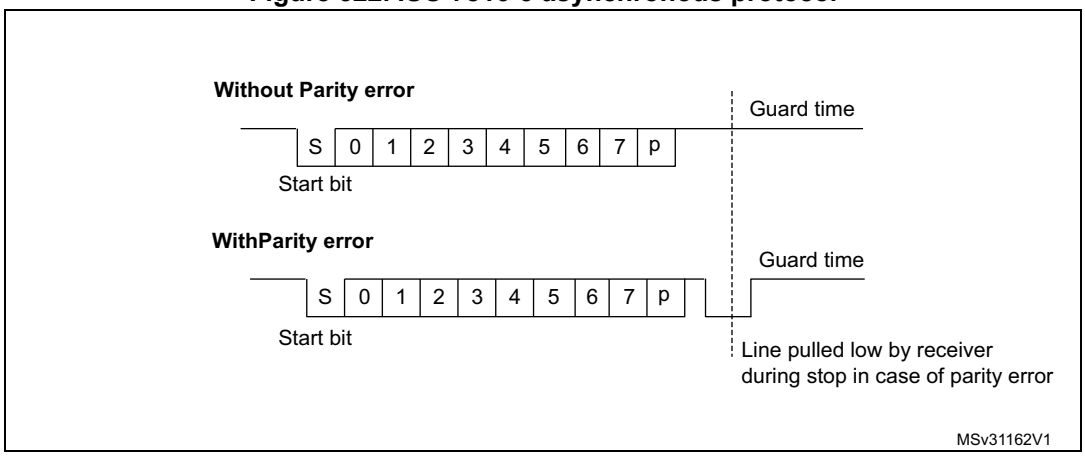
The USART must be configured as:

- 8 bits plus parity: M = 1 and PCE = 1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving data: STOP = 11 in the USART_CR2 register. It is also possible to choose 0.5 stop bit for reception.

In T = 0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

Figure 322 shows examples of what can be seen on the data line with and without parity error.

Figure 322. ISO 7816-3 asynchronous protocol



When connected to a smartcard, the TX output of the USART drives a bidirectional line that is also driven by the smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol.

The number of retries is programmed in the SCARCNT bitfield. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART_RQR register.

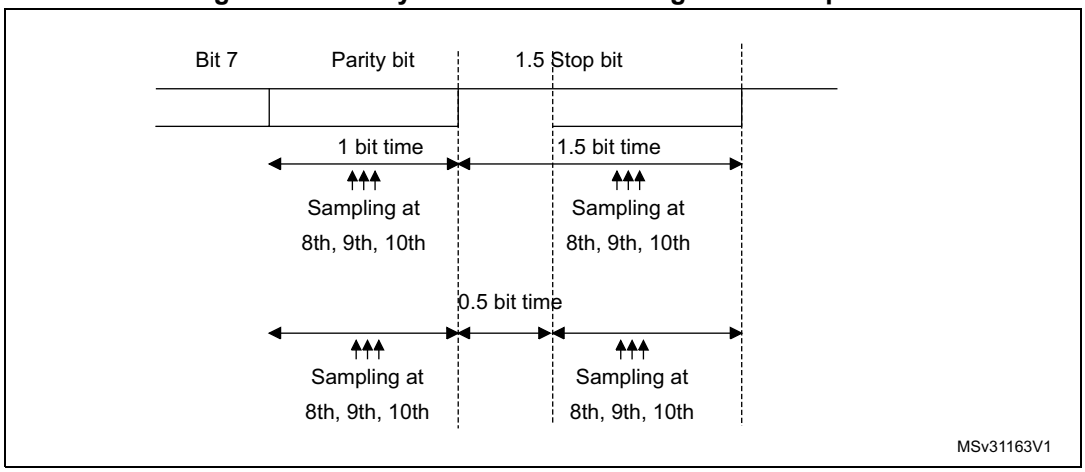
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guard time). If the software wants to repeat it again, it must ensure the minimum 2-baud periods required by the standard.
- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T = 1 mode). If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bitfield, the USART stops transmitting the NACK and signals the error as a parity error.
- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn't repeat the character.
- In transmission, the USART inserts the guard time (as programmed in the guard time register) between two successive characters. As the guard time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT character guard time, as defined by the 7816-3 specification) minus 12 (the duration of one character).
- The assertion of the TC flag can be delayed by programming the guard time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In smartcard mode an empty transmit shift register triggers the guard time counter to count up to the programmed value in the guard time register. TC is forced low during this time. When the guard time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.
- The de-assertion of TC flag is unaffected by smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

Note: Break characters are not significant in smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.

Figure 323 shows how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Figure 323. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the smartcard through the CK output. In smartcard mode, CK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART_GTPR register. CK frequency can be programmed from usart_ker_ck_pres/2 to usart_ker_ck_pres/62, where usart_ker_ck_pres is the peripheral input clock divided by a programmed prescaler.

Block mode (T = 1)

In T = 1 (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the USART_CR3 register.

When requesting a read from the smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) – 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt is generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

Note: The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time – 11 value), in order to enable the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baud time units. If the smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART signals it to the software through the RTOF flag and interrupt (when RTOIE bit is set).

Note: As in the smartcard protocol definition, the BWT/CWT values must be defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT – 11 or CWT – 11, respectively, taking into account the length of the last character itself.

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting. The length of the block is communicated by the smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value

(0x0). With this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value is programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN=LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN=LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

Note: The error checking code (LRC/CRC) must be computed/verified by software.

Direct and inverse convention

The smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 0, DATAINV = 0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 1, DATAINV =1.

Note: When logical data values are inverted (0 = H, 1 = L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH ≥ the USART received character is equal to 03 and the parity is odd.

Therefore, two methods are available for TS pattern recognition:

Method 1

The USART is programmed in standard smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card did not answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it is correctly received this time, by the reprogrammed USART

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

Method 2

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) LHHL LLL LLH = 0x103: inverse convention to be chosen

(H) LHHL HHH LLH = 0x13B: direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

36.5.19 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to *Section 36.4: USART implementation*.

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies the use of a return-to-zero, inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 324*).

The encoding and decoding of data in IrDA mode is handled by hardware blocks.

Even if IrDA is a half-duplex protocol, it requires two pins for transmission and reception (USART_TX and USART_RX).

The SIR transmit encoder modulates the non return-to-zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 kbauds for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is

ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not encoded. While receiving data, transmission must be avoided as the data to be transmitted may be corrupted.

- A 0 is transmitted as a high pulse and a 1 is transmitted as a 0. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see *Figure 325*).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 µs. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than two periods, where the period equals PSC / usart_ker_ck_pres (PSC being the prescaler value programmed in the PSC[7:0] bitfield of the USART_GTPR register). Pulses of width less than one period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than two periods are accepted as a pulse. The IrDA encoder/decoder does not work when PSC = 0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the stop bits in the USART_CR2 register must be configured to 1 stop bit.

IrDA low-power mode

- Transmitter

  In low-power mode, the pulse width is not maintained at 3 / 16 of the bit period. Instead, the width of the pulse is three times the IrDA low-power period. The IrDA low-power frequency minimum value is 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < IrDA low-power frequency < 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.
- Receiver

  Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART must discard pulses of duration shorter than one IrDA low-power period. A valid low is accepted only if its duration is greater than two IrDA low-power periods.

Note: IrDA low-power period = PSC / usart_ker_ck_pres, where PSC corresponds to the value programmed in the PSC[7:0] bitfield of *the USART_GTPR register.*

A pulse of width less than two and greater than one IrDA low-power period may or may not be rejected.

The receiver set-up time must be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception.

Figure 324. IrDA SIR ENDEC block diagram

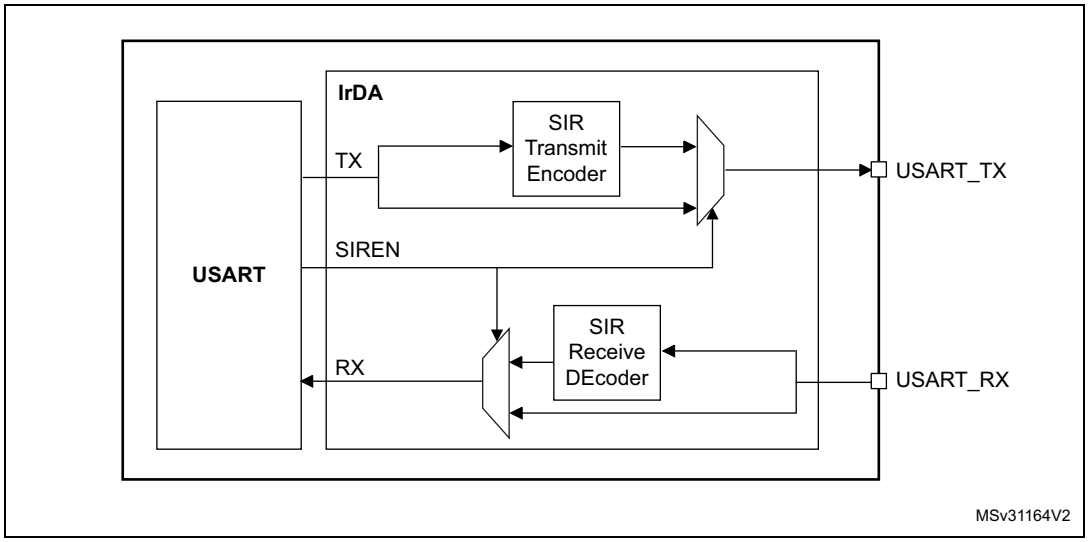
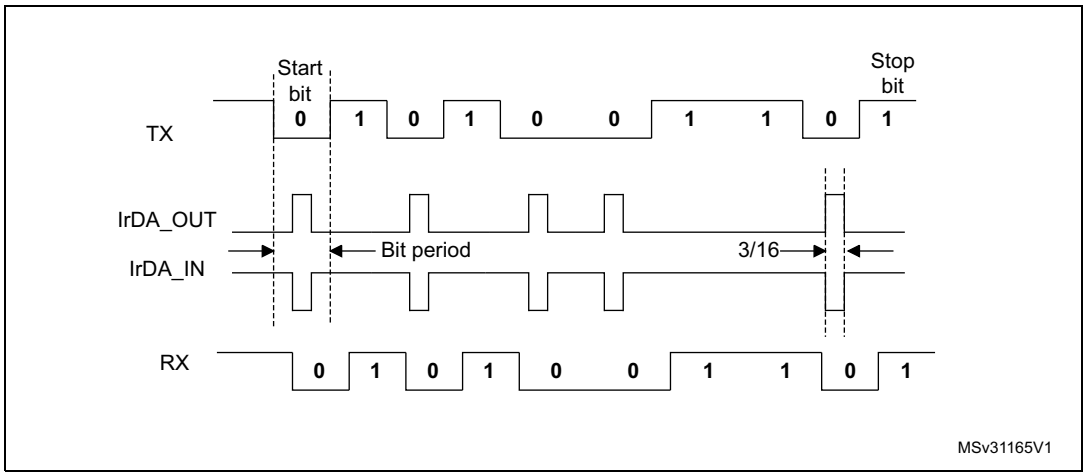


Figure 325. IrDA data modulation (3/16) - normal mode



36.5.20 Continuous communication using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 36.4: USART implementation to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 36.5.7. To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to *section direct memory access controller (DMA)*) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

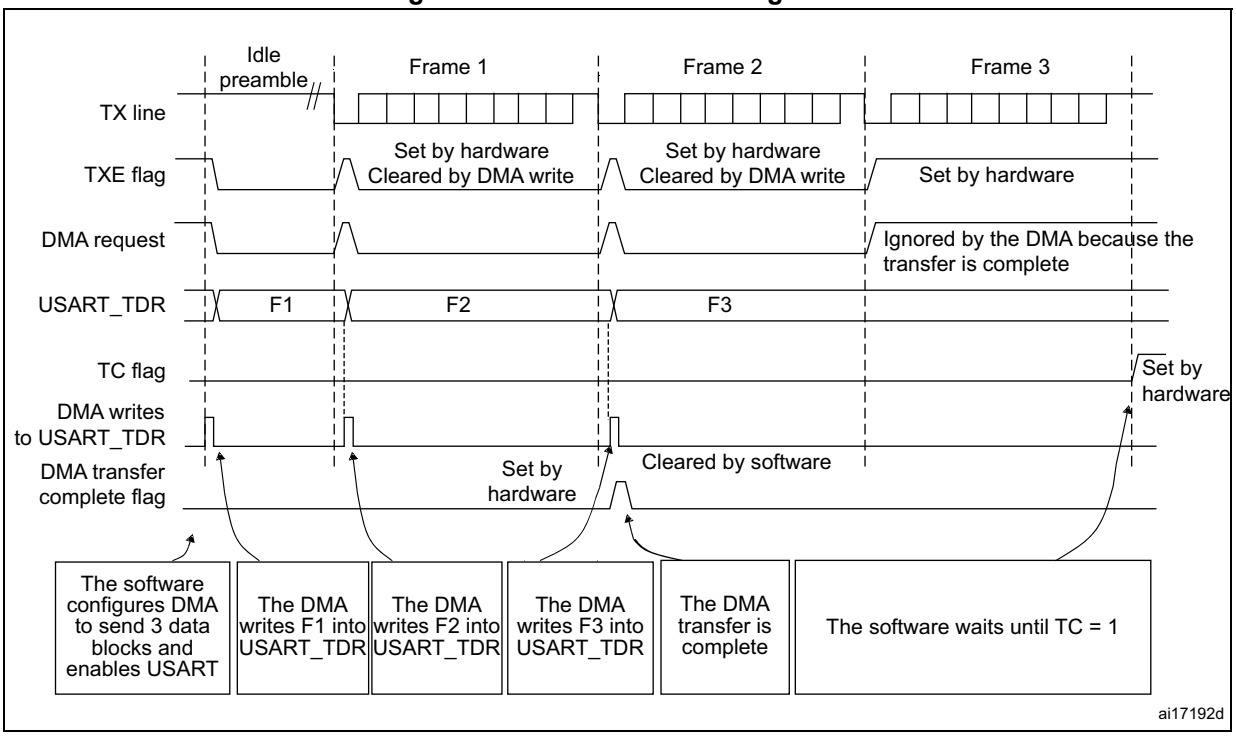
1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (DMA transfer complete), the TC flag can be monitored to make sure that the USART communication has completed. This is required to avoid corrupting the last transmission before disabling the USART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

Figure 326. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

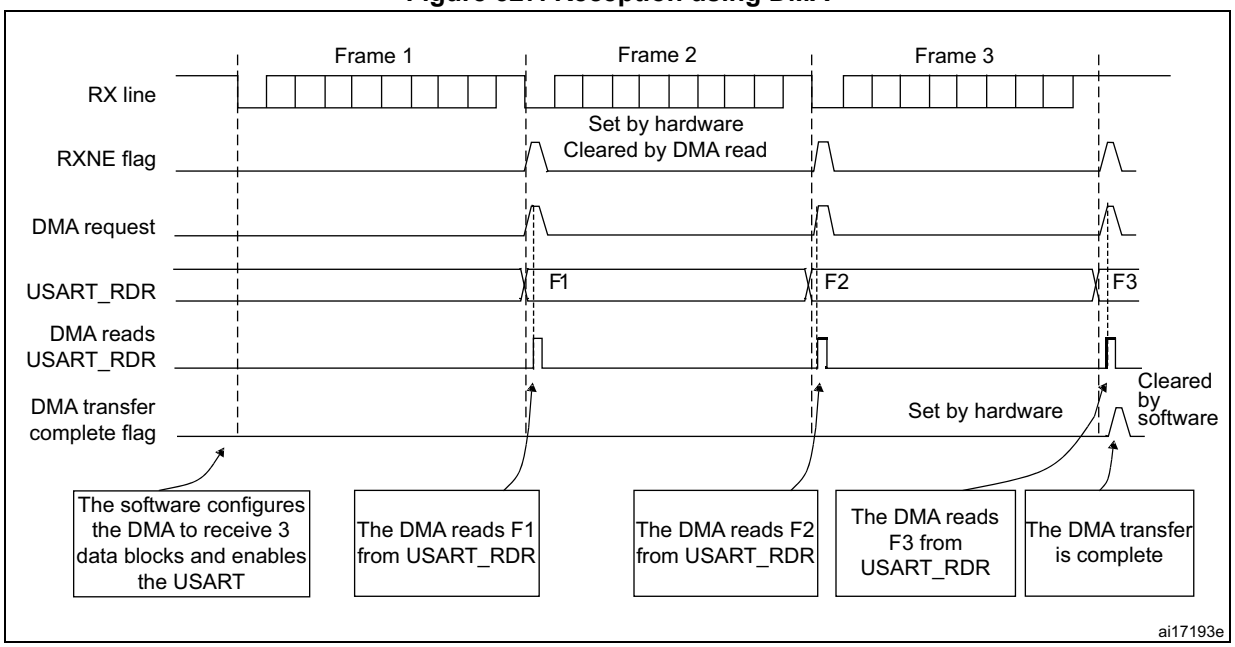
DMA mode can be enabled for reception by setting the DMAR bit in the USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to *section direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 327. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by receive FIFO not empty (that is, RXFNE = 1).

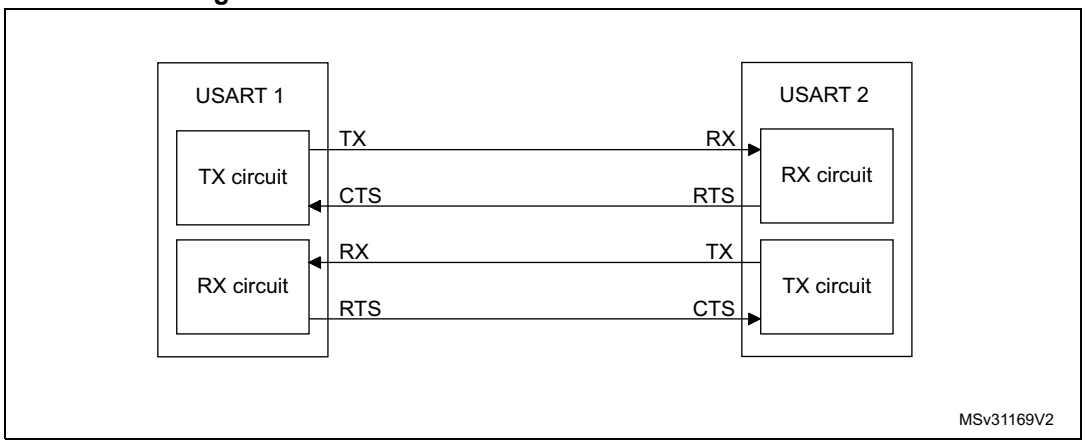
Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag, which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

36.5.21 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. The *Figure 328* shows how to connect two devices in this mode:

Figure 328. Hardware flow control between two USARTs

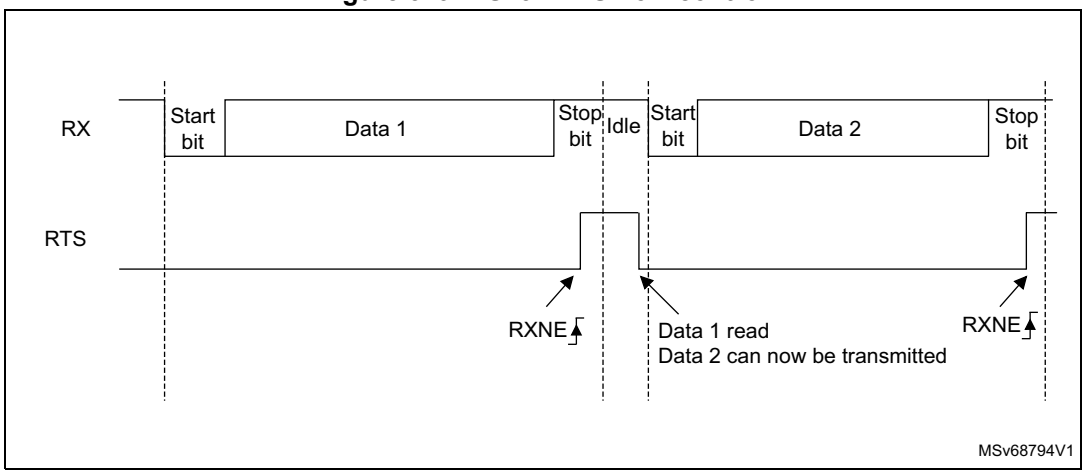


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to 1 in the USART_CR3 register.

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 329* shows an example of communication with RTS flow control enabled.

Figure 329. RS232 RTS flow control



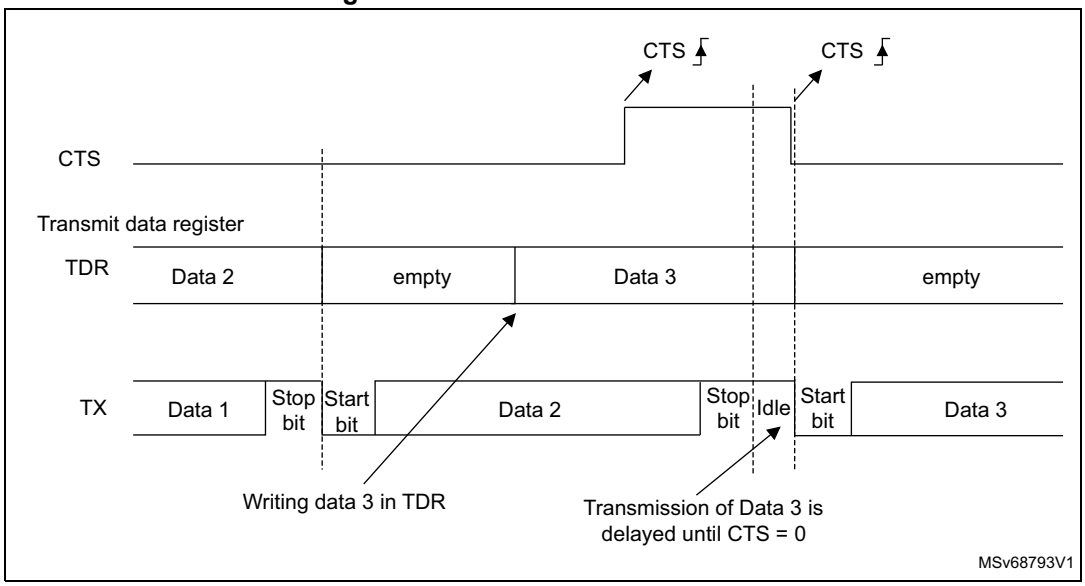
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission completes before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. *Figure 330* shows an example of communication with CTS flow control enabled.

Figure 330. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least three USART clock source periods before the end of the current character. In addition, it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This enables the user to activate the external transceiver control, through the DE (driver enable) signal. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the USART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

36.5.22 USART autonomous mode

The USART peripheral can be functional in Stop mode thanks to the autonomous mode. This mode can also be used in Run and Sleep mode. The UESM bit must be set prior to entering low-power mode.

The APB clock is requested by the peripheral each time the USART status needs to be updated. Once the USART receives the APB clock, it generates either an interrupt or a DMA request, depending on the peripheral configuration.

If an interrupt is generated, the device wakes up from Stop mode. If no interrupt is generated, the device remains in Stop mode but the kernel and APB clocks are still available for the USART and all the autonomous peripherals enabled in the reset and clock controller (RCC). If DMA requests are enabled, the data are directly transferred to/from the SRAM thanks to the DMA while the product remains in Stop mode.

USART transmission mode

In transmission, the APB clock is requested only when the TE bit is set and in the following cases:

- If the FIFO mode is enabled, the APB clock is requested when
  - The TxFIFO is empty (TXFE = 1) and the corresponding interrupt is enabled (TXFEIE = 1)
  - The TxFIFO threshold is reached (TXFT = 1) and the corresponding interrupt is enabled (TXFTIE = 1)
  - The TxFIFO is not full (TXFNF = 1) and the corresponding interrupt or DMA is enabled (TXFNFIE = 1 or DMAT = 1)
- If the FIFO mode is disabled, the APB clock is requested as soon as data are transferred to the shift register. The DMA or associated interrupt must be enabled.

The TE bit is set by hardware if an asynchronous trigger is detected.

A transmission is automatically launched when an asynchronous trigger is detected in Run, Sleep, or Stop mode. The trigger is selected through the TRIGSEL bit in the USART_AUTOCR register. It sets the TE bit in the USART_CR1 register and generates an APB clock request to enable the transfer. The APB clock is requested until the transmission completes and the TE bit is cleared by hardware when the programmed number of data to be transmitted (TDN bitfield in the USART_AUTOCR register) is reached. In this case, the TC flag is set when the number of data to be transmitted is reached and the last byte is transmitted.

USART reception mode

- If the FIFO mode is enabled, the APB clock is requested when
  - The RxFIFIO is full (RXFF = 1) and the corresponding interrupt is enabled (RXFFIE = 1)
  - The RxFIFO threshold is reached (RXFT = 1) and the corresponding interrupt is enabled (RXFTIE = 1)
  - The RxFIFO is not empty (RXFNE = 1) and the corresponding interrupt or DMA is enabled (RXFNEIE= 1 or DMAR = 1)
- If the FIFO mode is disabled, the APB clock is requested when the USART finishes sampling data and it is ready to be written in the USART_RDR. The DMA or the associated interrupt must be enabled.

Note: The APB clock is requested in reception mode when an overrun error occurs (ORE = 1). The EIE bit must be set to enable the generation of an interrupt and waking up the MCU, and the OVRDIS bit must remain cleared. The APB clock request is kept until the interrupt flag is cleared.

The APB clock is also requested in reception mode when a Parity/Noise/Framing error occurs and the DMA is used for reception. The APB clock request is kept until the interrupt flag is cleared.

Only UART and SPI master modes support the autonomous mode.

Determining the maximum USART baud rate that enables to correctly wake up the microcontroller from low-power mode

The maximum baud rate that enables to correctly wake up the microcontroller from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the USART receiver tolerance (see *Section 36.5.9: Tolerance of the USART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = 01, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 348: Tolerance of the USART receiver when BRR [3:0] = 0000*, the USART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < USART receiver tolerance

$D_{WUmax} = t_{WUUSART} / (11 \times T_{bitmin})$

$T_{bitmin} = t_{WUUSART} / (11 \times D_{WUmax})$

where $t_{WUUSART}$ is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC, and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In fact, we need to consider at least the usart_ker_ck inaccuracy (DREC).

For example, if HSI is used as usart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WUUSART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = USART receiver tolerance – DREC = 3.41% – 1% = 2.41%

T_{bitmin} = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate enables to wake up correctly from low-power mode is: 1/11.32 µs = 88.36 kbauds.

36.6 USART in low-power modes

Table 351. Effect of low-power modes on the USART

Mode	Description
Sleep	No effect. USART interrupts cause the device to exit Sleep mode.
Stop 0 and Stop 1(1)	The content of the USART registers is kept. If the USART is clocked by an oscillator available in Stop mode, transfers in asynchronous and SPI master modes are functional. DMA requests are functional, and the interrupts cause the device to exit Stop mode.
Stop 2 and Stop 3	The USART peripheral is powered down and must be reinitialized after exiting Stop mode.
Standby	The USART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 36.4: USART implementation* to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

36.7 USART interrupts

Refer to *Table 352* for a detailed description of all USART interrupt requests.

Table 352. USART interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop modes(1)	Exit from Standby mode
USART or UART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	Yes	No
	Transmit FIFO not full	TXFNF	TXFNFIE	TXFIFO full		Yes	
	Transmit FIFO empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR(2)		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission complete	TC	TCIE	Write TDR or write 1 in TCCF		Yes	
	Transmission complete Before guard time	TCBGT	TCBGTIE	Write TDR or write 1 in TCBGT		No	

Table 352. USART interrupt requests (continued)

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop modes(1)	Exit from Standby mode
USART or UART	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	No
	Receive FIFO not empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO full	RXFF(3)	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RX-NEIE/RX FNEIE	Write 1 in ORECF		Yes	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		Yes(4)	
	LIN break	LBDF	LBDIE	Write 1 in LBDCF		No	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		Yes(4)	
	Overrun error in multibuffer communication.	ORE(5)		Write 1 in ORECF		Yes	
	Framing error in multibuffer communication.	FE		Write 1 in FECF		Yes(4)	
	Character match	CMF	CMIE	Write 1 in CMCF		Yes(6)	
	Receiver timeout	RTOF	RTOFIE	Write 1 in RTOCCF		No	
	End of Block	EOBF	EOBIE	Write 1 in EOBCF		No	
	SPI slave underrun error	UDR	EIE	Write 1 in UDRCF		No	

1. The USART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to *Section 36.4: USART implementation* for the list of supported Stop modes.
2. Writing to TDR clears the TXFT flag only if the number of empty locations is less than the value programmed in TXFTCFG[2:0].
3. RXFF flag is asserted if the USART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in USART_RDR. In Stop mode, USART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
4. Parity/Noise/Framing error interrupts enable waking up from Stop modes when the DMA is used.
5. When OVRDIS = 0.
6. The DMA must be used when the FIFO mode is enabled.

36.8 USART registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

36.8.1 USART control register 1 (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enable, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE**: RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when RXFF = 1 in the USART_ISR register

Bit 30 **TXFEIE**: TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFE = 1 in the USART_ISR register

Bit 29 **FIFOEN**: FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length
This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
M[1:0] = 00: 1 start bit, 8 Data bits, n stop bits
M[1:0] = 01: 1 start bit, 9 Data bits, n stop bits
M[1:0] = 10: 1 start bit, 7 Data bits, n stop bits
This bit can only be written when the USART is disabled (UE = 0).
Note: In 7-bits data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 36.4: USART implementation.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method
This bit determines the USART wake-up method from mute mode. It is set or cleared by software.
0: Idle line
1: Address mark
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable
This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M=0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).
0: Parity control disabled
1: Parity control enabled
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.
0: Even parity
1: Odd parity
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TXFNF = 1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever ORE = 1 or RXFNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. When the autonomous mode is not used, TE bit is set and cleared by software. When the autonomous mode is used, TE bit becomes a status bit, which is set and cleared by hardware.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the USART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot request its kernel clock and is not functional in low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not functional in low-power mode.

1: USART functional in low-power mode.

Note: The UESM bit must be set at the initialization phase.

If the USART does not support the wake-up from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

36.8.2 USART control register 1 [alternate] (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN**: FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length
This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
M[1:0] = 00: 1 start bit, 8 Data bits, n stop bits
M[1:0] = 01: 1 start bit, 9 Data bits, n stop bits
M[1:0] = 10: 1 start bit, 7 Data bits, n stop bits
This bit can only be written when the USART is disabled (UE = 0).
Note: In 7-bits data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of block interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the EOBF flag is set in the USART_ISR register
Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 26 **RTOIE**: Receiver timeout interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.
Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 36.4: USART implementation.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the USART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TXE = 1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: USART interrupt is generated whenever ORE = 1 or RXNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. When the autonomous mode is not used, TE bit is set and cleared by software. When the autonomous mode is used, TE bit becomes a status bit, which is set and cleared by hardware.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the USART acts as a transmitted, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot request its kernel clock and is not functional in low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not functional in low-power mode.

1: USART functional in low-power mode.

Note: The UESM bit must be set at the initialization phase.

If the USART does not support the wake-up from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

36.8.3 USART control register 2 (USART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw

Bits 31:24 **ADD[7:0]**: Address of the USART node

These bits give the address of the USART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter must be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When a character, received during low-power mode, corresponds to the character programmed through ADD[7:0] bitfield, the CMF flag is set and wakes up the device from low-power mode if the corresponding interrupt is enabled by setting CMIE bit.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the USART is disabled (UE = 0).

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.

0: Receiver timeout feature disabled.

1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bits 22:21 **ABRMOD[1:0]**: Auto baud rate mode

These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.

01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 -> Frame = Start10xxxxxx)

10: 0x7F frame detection.

11: 0x55 frame detection

This bitfield can only be written when ABREN = 0 or the USART is disabled (UE = 0).

Note: If DATAINV = 1 and/or MSBFIRST = 1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)

If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 20 **ABREN**: Auto baud rate enable

This bit is set and cleared by software.

0: Auto baud rate detection is disabled.

1: Auto baud rate detection is enabled.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1 = H, 0 = L)

1: Logical data from the data register are send/received in negative/inverse logic. (1 = L, 0 = H). The parity bit is also inverted.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)

1: TX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).

This enables the use of an external inverter on the TX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)

1: RX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).

This enables the use of an external inverter on the RX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 14 **LINEN**: LIN mode enable

This bit is set and cleared by software.

0: LIN mode disabled

1: LIN mode enabled

The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the USART_CR1 register, and to detect LIN Sync breaks.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support LIN mode, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bits 13:12 **STOP[1:0]**: Stop bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: 0.5 stop bit.

10: 2 stop bits

11: 1.5 stop bits

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 11 **CLKEN**: Clock enable

This bit enables the user to enable the CK pin.

0: CK pin disabled

1: CK pin enabled

Note: If neither synchronous mode nor smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

In smartcard mode, in order to provide correctly the CK clock to the smartcard, the steps below must be respected:
UE = 0
SCEN = 1
GTPR configuration
CLKEN= 1
UE = 1

Bit 10 **CPOL**: Clock polarity

This bit enables the user to select the polarity of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on CK pin outside transmission window

1: Steady high value on CK pin outside transmission window

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see *Figure 312* and *Figure 313*)

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the CK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the CK pin

1: The clock pulse of the last data bit is output to the CK pin

Caution: The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART_CR1 register.

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable

Break interrupt mask (break detection using break delimiter).

0: Interrupt is inhibited

1: An interrupt is generated whenever LBDF = 1 in the USART_ISR register

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 5 **LBDL**: LIN break detection length

This bit is for selection between 11 bit or 10 bit break detection.

0: 10-bit break detection

1: 11-bit break detection

This bit can only be written when the USART is disabled (UE = 0).

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 4 **ADDM7**: 7-bit address detection/4-bit address detection

This bit is for selection between 4-bit address detection or 7-bit address detection.

0: 4-bit address detection

1: 7-bit address detection (in 8-bit data mode)

This bit can only be written when the USART is disabled (UE = 0)

Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bit 3 **DIS_NSS**:

When the DIS_NSS bit is set, the NSS pin input is ignored.

0: SPI slave selection depends on NSS input pin.

1: SPI slave is always selected and NSS input pin is ignored.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **SLVEN**: Synchronous slave mode enable

When the SLVEN bit is set, the synchronous slave mode is enabled.

0: Slave mode disabled.

1: Slave mode enabled.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Note: The CPOL, CPHA, and LBCL bits must not be written while the transmitter is enabled.

36.8.4 USART control register 3 (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXF TIE	RXFTCFG[2:0]			TCBG TIE	TXFTIE	Res.	Res.	Res.	SCARCNT[2:0]			Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HD SEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth
001:TXFIFO reaches 1/4 of its depth
010:TXFIFO reaches 1/2 of its depth
011:TXFIFO reaches 3/4 of its depth
100:TXFIFO reaches 7/8 of its depth
101:TXFIFO becomes empty
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG[2:0].

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth
001:Receive FIFO reaches 1/4 of its depth
010:Receive FIFO reaches 1/2 of its depth
011:Receive FIFO reaches 3/4 of its depth
100:Receive FIFO reaches 7/8 of its depth
101:Receive FIFO becomes full
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TCBGT = 1 in the USART_ISR register
Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG[2:0].

Bits 22:20 Reserved, must be kept at reset value.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count
This bitfield specifies the number of retries for transmission and reception in smartcard mode.
In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).
In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).
This bitfield must be programmed only when the USART is disabled (UE = 0).
When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.
0x0: retransmission disabled - No automatic retransmission in transmission mode.
0x1 to 0x7: number of automatic retransmission attempts (before signaling error)
Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 36.4: USART implementation on page 1269.

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for smartcard mode)

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE(RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data are written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling smartcard mode.

0: Smartcard mode disabled

1: Smartcard mode enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half-duplex mode is not selected

1: Half-duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1 or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

36.8.5 USART control register 3 [alternate] (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBGTIE	Res.	Res.	Res.	Res.	SCARCNT[2:0]			Res.
							rw					rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TCBGT = 1 in the USART_ISR register

Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 Reserved, must be kept at reset value.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count

This bitfield specifies the number of retries for transmission and reception in smartcard mode.

In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).

This bitfield must be programmed only when the USART is disabled (UE = 0).

When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.

0x0: retransmission disabled - No automatic retransmission in transmission mode.

0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 36.4: USART implementation.

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for smartcard mode)

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE(RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data are written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling smartcard mode.

0: Smartcard mode disabled

1: Smartcard mode enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half-duplex mode is not selected

1: Half-duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

36.8.6 USART baud rate register (USART_BRR)

This register can only be written when the USART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRR[15:0]:** USART baud rate

BRR[15:4]

BRR[15:4] correspond to USARTDIV[15:4]

BRR[3:0]

When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].

When OVER8 = 1:

BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.

BRR[3] must be kept cleared.

36.8.7 USART guard time and prescaler register (USART_GTPR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT[7:0]								PSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **GT[7:0]**: Guard time value

This bitfield is used to program the Guard time value in terms of number of baud clock periods.

This is used in smartcard mode. The transmission complete flag is set after this guard time value.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bits 7:0 **PSC[7:0]**: Prescaler value

Condition: IrDA low-power and normal IrDA mode

PSC[7:0] = IrDA normal and Low-power baud rate

This bitfield is used for programming the prescaler for dividing the USART source clock to achieve the low-power frequency:

The source clock is divided by the value given in the register (8 significant bits):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

Condition: Smartcard mode

PSC[4:0]: Prescaler value

This bitfield is used for programming the prescaler for dividing the USART source clock to provide the smartcard clock.

The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

This bitfield can only be written when the USART is disabled (UE = 0).

Note: Bits [7:5] must be kept cleared if smartcard mode is used.

This bitfield is reserved and forced by hardware to 0 when the smartcard and IrDA modes are not supported. Refer to Section 36.4: USART implementation.

36.8.8 USART receiver timeout register (USART_RTOR)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bits 31:24 **BLEN[7:0]**: Block Length

This bitfield gives the block length in smartcard T = 1 reception. Its value equals the number of information characters + the length of the Epilogue Field (1 – LEC / 2 – CRC) – 1.

Examples:

BLEN = 0 -> 0 information characters + LEC

BLEN = 1 -> 0 information characters + CRC

BLEN = 255 -> 254 information characters + CRC (total 256 characters))

In smartcard mode, the block length counter is reset when TXE = 0 (TXFE = 0 in case FIFO mode is enabled).

This bitfield can be used also in other modes. In this case, the block length counter is reset when RE = 0 (receiver disabled) and/or when the EOBCF bit is written to 1.

Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bitfield gives the Receiver timeout value in terms of number of bit duration.

In Standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In smartcard mode, this value is used to implement the CWT and BWT. See smartcard chapter for more details. In the standard, the CWT/BWT measurement is done starting from the start bit of the last received character.

Note: This value must only be programmed once per received character.

Note: RTOR can be written on-the-fly. If the new value is lower than or equal to the counter, the RTOF flag is set.

This register is reserved and forced by hardware to "0x00000000" when the receiver timeout feature is not supported. Refer to Section 36.4: USART implementation.

36.8.9 USART request register (USART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

When FIFO mode is disabled, writing 1 to this bit sets the TXE flag. This enables to discard the transmit data. This bit must be used only in smartcard mode, when data have not been sent due to errors (NACK) and the FE flag is active in the USART_ISR register. If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value.

When FIFO is enabled, TXFRQ bit is set to flush the whole FIFO. This sets the TXFE flag (Transmit FIFO empty, bit 23 in the USART_ISR register). Flushing the Transmit FIFO is supported in both UART and smartcard modes.

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit empties the entire receive FIFO, that is clears the bit RXFNE.

This enables to discard the received data without reading them, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the USART in mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: When the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 **ABRRQ**: Auto baud rate request

Writing 1 to this bit resets the ABRF and ABRE flags in the USART_ISR and requests an automatic baud rate measurement on the next received data frame.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

36.8.10 USART interrupt and status register (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY
				r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of USART_CR3 register.
An interrupt is generated if the TXFTIE bit (bit 31) is set in the USART_CR3 register.
0: TXFIFO does not reach the programmed threshold.
1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the threshold programmed in the RXFTCFG[2:0] bitfield of the USART_CR3 register is reached. This means that there are (RXFTCFG[2:0] – 1) data in the Receive FIFO and one data in the USART_RDR register. An interrupt is generated if the RXFTIE bit = 1 (bit 28) in the USART_CR3 register.
0: Receive FIFO does not reach the programmed threshold.
1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG[2:0] threshold is configured to 101, the RXFT flag is set if RXFIFO size data are available, that is, (RXFIFO size – 1) data in the RXFIFO and 1 data in the USART_RDR. Consequently, the (RXFIFO size + 1) th received data does not cause an overrun error. The overrun error occurs after receiving the (RXFIFO size + 2) th data.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.
It is set by hardware in smartcard mode, if the transmission of a frame containing data has completed and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.
This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.
0: Transmission has not completed, or transmission has completed unsuccessfully (that is, a NACK is received from the card)
1: Transmission has completed successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 36.4: USART implementation.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART_RDR register.
An interrupt is generated if the RXFFIE bit = 1 in the USART_CR1 register.
0: RXFIFO not full.
1: RXFIFO Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART_RQR register.
An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the USART_CR1 register.
0: TXFIFO not empty.
1: TXFIFO empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 36.4: USART implementation.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up on idle mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 36.4: USART implementation.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1in the USART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception ongoing

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXFNE is also set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation has completed without success (ABRE = 1) (ABRE, RXFNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 36.4: USART implementation.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to EOBCF in the USART_ICR register.

0: End of block not reached

1: End of block (number of characters) reached

Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 36.4: USART implementation.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 36.4: USART implementation.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the USART_TDR. Every write operation to the USART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART_TDR.

An interrupt is generated if the TXFNFIE bit = 1 in the USART_CR1 register.

0: Transmit FIFO is full

1: Transmit FIFO is not full

Note: The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE is set at the same time).

This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.

The TC flag behaves as follows:

- When TDN = 0, the TC flag is set when the transmission of a frame containing data has completed and when TXE/TXFE is set.
- When TDN is equal to the number of data in the TXFIFO, the TC flag is set when TXFIFO is empty and TDN is reached.
- When TDN is greater than the number of data in the TXFIFO, TC remains cleared until the TXFIFO is filled again to reach the programmed number of data to be transferred.
- When TDN is less than the number of data in the TXFIFO, TC is set when TDN is reached even if the TXFIFO is not empty.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

The TC bit is cleared by software, by writing 1 to the TCCF of the USART_ICR register, or by writing to the USART_TDR register.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART_RDR register. Every read operation from the USART_RDR frees a location in the RXFIFO.

RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 36.5.9: Tolerance of the USART receiver to clock deviation).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the USART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.
An interrupt is generated if PEIE = 1 in the USART_CR1 register.
0: No parity error
1: Parity error

Note: This error is associated with the character in the USART_RDR.

36.8.11 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported)

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	RE ACK	TE ACK	Res.	RWU	SBKF	CMF	BUSY
						r			r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.

It is set by hardware in smartcard mode, if the transmission of a frame containing data has completed, and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission has not completed or transmission has completed unsuccessfully (that is, a NACK is received from the card)

1: Transmission has not completed successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 36.4: USART implementation.

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 36.4: USART implementation.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up on idle mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 36.4: USART implementation.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1 in the USART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception ongoing

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXNE is also set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation has completed without success (ABRE = 1) (ABRE, RXNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 36.4: USART implementation.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to EOBCF in the USART_ICR register.

0: End of block not reached

1: End of block (number of characters) reached

Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 36.4: USART implementation.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 36.4: USART implementation.

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by writing to the USART_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in smartcard T = 0 mode, in case of transmission failure).

An interrupt is generated if the TXEIE bit = 1 in the USART_CR1 register.

0: Data register full

1: Data register empty

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the USART_ICR register or by writing to the USART_TDR register.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the USART_RDR shift register has been transferred to the USART_RDR register. It is cleared by reading from the USART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXNE = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 36.5.9: Tolerance of the USART receiver to clock deviation).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error

1: Parity error

Note: This error is associated with the character in the USART_RDR.

36.8.12 USART interrupt flag clear register (USART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMCF	Res.
														w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
		w	w	w		w	w	w	w	w	w	w	w	w	w

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the USART_ISR register.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **UDRCF**:SPI slave underrun clear flag

Writing 1 to this bit clears the UDRF flag in the USART_ISR register.

Note: If the USART does not support SPI slave mode, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation

Bit 12 **EOBCF**: End of block clear flag

Writing 1 to this bit clears the EOBF flag in the USART_ISR register.

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 11 **RTOCF**: Receiver timeout clear flag

Writing 1 to this bit clears the RTOF flag in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag
Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.
Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 8 **LBDCF**: LIN break detection clear flag
Writing 1 to this bit clears the LBDF flag in the USART_ISR register.
Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

Bit 7 **TCBGTCF**: Transmission complete before Guard time clear flag
Writing 1 to this bit clears the TCBGT flag in the USART_ISR register.

Bit 6 **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC flag in the USART_ISR register.

Bit 5 **TXFECF**: TXFIFO empty clear flag
Writing 1 to this bit clears the TXFE flag in the USART_ISR register.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the USART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the USART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the USART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the USART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the USART_ISR register.

36.8.13 USART receive data register (USART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see *Section 36.5.1: USART block diagram*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

36.8.14 USART transmit data register (USART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The USART_TDR register provides the parallel interface between the internal bus and the output shift register (see *Section 36.5.1: USART block diagram*).

When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF = 1.

36.8.15 USART prescaler register (USART_PRESC)

This register can only be written when the USART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The USART input clock can be divided by a prescaler factor:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Others: Reserved, must not be used

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011 that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation.

36.8.16 USART autonomous mode control register (USART_AUTOCR)

Address offset: 0x30

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGSEL[3:0]				IDLEDIS	TRIGEN	TRIGPOL
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:19 **TRIGSEL[3:0]**: Trigger selection bits

Refer to *Description of USART interconnections*.
This bitfield can be written only when the UE bit is cleared in USART_CR1 register.
0000: usart_trg0 selected
0001: usart_trg1 selected
...
1111: usart_trg15 selected

Note: This bitfield can be written only when the UE bit of USART_CR1 register is cleared.

Bit 18 **IDLEDIS**: Idle frame transmission disable bit after enabling the transmitter

0: Idle frame sent after enabling the transmitter (TE = 1 in USART_CR1)
1: Idle frame not sent after enabling the transmitter

Note: This bitfield can be written only when the UE bit of USART_CR1 register is cleared.

Bit 17 **TRIGEN**: Trigger enable bit

0: Trigger disabled

1: Trigger enabled

Note: This bitfield can be written only when the UE bit of USART_CR1 register is cleared.

When a trigger is detected, TE is set to 1 in USART_CR1 and the data transfer is launched.

Bit 16 **TRIGPOL**: Trigger polarity bit

This bitfield can be written only when the UE bit is cleared in USART_CR1 register.

0: Trigger active on rising edge

1: Trigger active on falling edge

Bits 15:0 **TDN[15:0]**: TDN transmission data number

This bitfield enables the programming of the number of data to be transmitted. It can be written only when UE is cleared in USART_CR1.

36.8.17 USART register map

Table 353. USART register map and reset values

Offset	Register name	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
0x00	USART_CR1 FIFO mode enabled	RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	USART_CR1 FIFO mode disabled	Res.	Res.	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	USART_CR2	ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0			0
0x08	USART_CR3 FIFO mode enabled	TXFTCFG[2:0]			RXFTIE	RXFTCFG2:0]			TCBGTIE	TXFTIE	Res.	Res.	Res.	SCAR CNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value	0	0	0	0	0	0	0	0	0				0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	USART_CR3 FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBGTIE	Res.	Res.	Res.	Res.	SCAR CNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value								0					0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USART_ GTPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 353. USART register map and reset values (continued)

Offset	Register name	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
0x14	USART_RTOR	BLEN[7:0]								RTO[23:0]																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	USART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
	Reset value																												0	0	0	0	0
0x1C	USART_ISR FIFO mode enabled	Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value					0	0	X	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x1C	USART_ISR FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value							0			0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x20	USART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMCF	Res.	Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
	Reset value															0				0	0	0		0	0	0	0	0	0	0	0	0	0
0x24	USART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x28	USART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x2C	USART_PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0
0x30	USART_AUTOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGSEL[3:0]				IDLEDIS	TRIGEN	TRIGPOL	TDN[15:0]															
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

37 Low-power universal asynchronous receiver transmitter (LPUART)

37.1 LPUART introduction

The LPUART is an UART, which enables bidirectional UART communications with a limited power consumption. Only a 32.768 kHz LSE clock is required to enable UART communications up to 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the microcontroller is in low-power mode, the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports half-duplex single-wire communications and modem operations (CTS/RTS).

It also supports multiprocessor communications.

DMA (direct memory access) can be used for data transmission/reception.

37.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 bauds to 9600 bauds using a 32.768 kHz clock source.
- Higher baud rates can be achieved by using a higher frequency clock source
- Two internal FIFOs to transmit and receive data

  Each FIFO can be enabled/disabled by software and come with status flags for FIFOs states.
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK.
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS485 transceiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - Busy and end of transmission flags

- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Four error detection flags:
  - Overrun error
  - Noise detection
  - Frame error
  - Parity error
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode
- Autonomous functionality in Stop mode

37.3 LPUART implementation

The tables below describe the LPUART implementation. USARTs and UARTs are included for comparison.

Table 354. Instance implementation on STM32WBA2

Instance	STM32WBA2
USART1	Full
LPUART1	Low-power

Table 355. USART/LPUART features

Modes/features(1)	Full feature set	Basic feature set	Low-power feature set
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (master/slave)	X	-	-
Smartcard mode	X	-	-
Single-wire half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual clock domain	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver enable	X	X	X

Table 355. USART/LPUART features (continued)

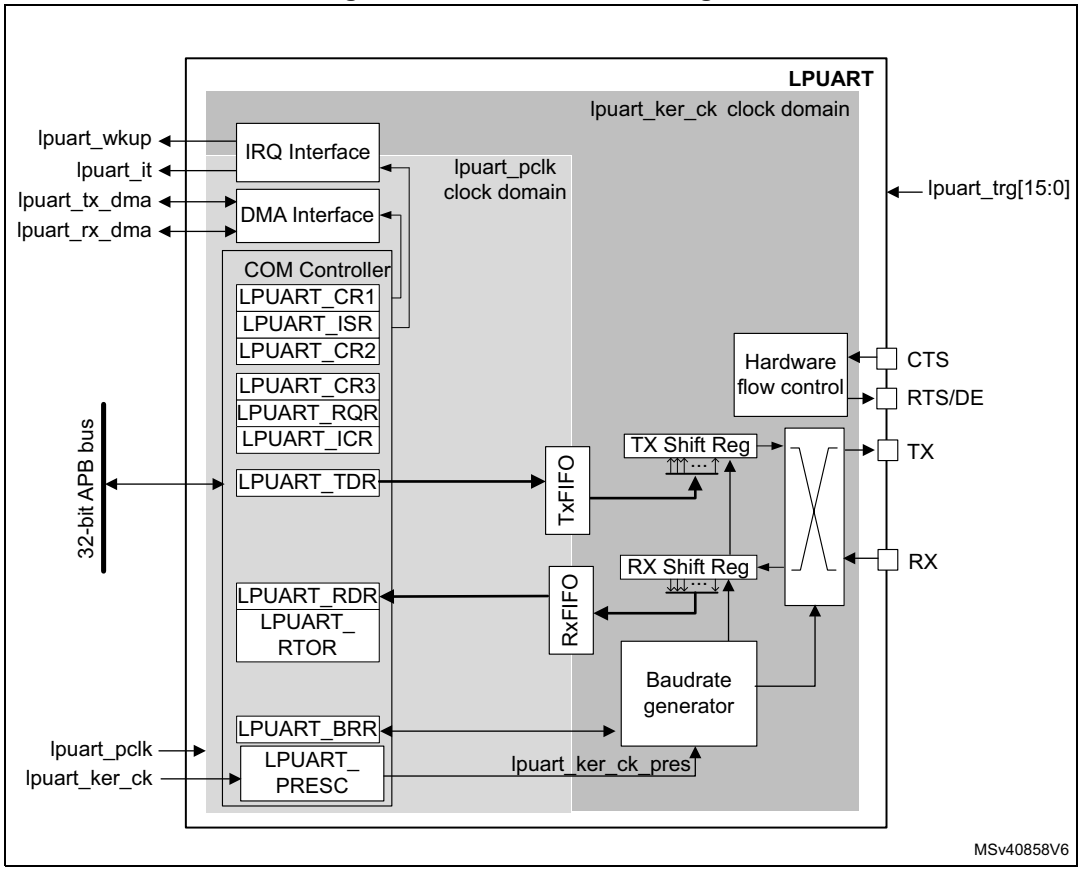
Modes/features(1)	Full feature set	Basic feature set	Low-power feature set
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	8		
Wake-up from low-power mode	X(2)	X(2)	X(3)
Autonomous mode	X	X	X

1. X = supported.
2. Wake-up supported from Stop 0 and Stop 1 modes.
3. Wake-up supported from Stop 0, Stop 1, and Stop 2 modes.

37.4 LPUART functional description

37.4.1 LPUART block diagram

Figure 331. LPUART block diagram



37.4.2 LPUART pins and internal signals

Description LPUART input/output pins

- LPUART bidirectional communications

  LPUART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

  – **RX** (Receive Data Input):

    RX is the serial data input.

  – **TX (**Transmit Data Output)

    When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire mode, this I/O is used to transmit and receive the data.

- RS232 hardware flow control mode

  The RS232 hardware flow control mode requires the following pins:

  – **CTS (**Clear To Send)

    When driven high, this signal blocks the data transmission at the end of the current transfer.

  – **RTS** (Request to send)

    When it is low, this signal indicates that the LPUART is ready to receive data.

- RS485 hardware flow control mode

  The **DE** (Driver Enable) pin is required in RS485 hardware control mode. This signal activates the transmission mode of the external transceiver.

Refer to *Table 356* and *Table 357* for the list of LPUART input/output pins and internal signals.

Table 356. LPUART input/output pins

Pin name	Signal type	Description
LPUART_RX	Input	Serial data receive input.
LPUART_TX	Output	Transmit data output.
LPUART_CTS	Input	Clear to send
LPUART_RTS	Output	Request to send
LPUART_DE(1)	Output	Driver enable

1. LPUART_DE and LPUART_RTS share the same pin.

Description LPUART input/output signals

Table 357. LPUART internal input/output signals

Signal name	Signal type	Description
lpuart_pclk	Input	APB clock
lpuart_ker_ck	Input	LPUART kernel clock
lpuart_wkup	Output	LPUART provides a wake-up interrupt

Table 357. LPUART internal input/output signals (continued)

Signal name	Signal type	Description
lpuart_it	Output	LPUART global interrupt
lpuart_tx_dma	Input/output	LPUART transmit DMA request
lpuart_rx_dma	Input/output	LPUART receive DMA request
lpuart_trg[15:0]	Input	LPUART triggers

Description LPUART interconnections

Table 358. LPUART interconnections (LPUART1)

Signal name	Source
lpuart_trg0	lpdma1_ch0_tc
lpuart_trg1	lpdma1_ch1_tc
lpuart_trg2	lpdma1_ch2_tc
lpuart_trg3	lpdma1_ch3_tc
lpuart_trg4	exti6
lpuart_trg5	exti8
lpuart_trg6	lptim1_ch1
lpuart_trg7	-
lpuart_trg8	-
lpuart_trg9	-
lpuart_trg10	rtc_alra_trg
lpuart_trg11	rtc_wut_trg
lpuart_trg12	-
lpuart_trg13	-
lpuart_trg14	-
lpuart_trg15	-

37.4.3 LPUART clocks

The simplified block diagram given in *Section 37.4.1: LPUART block diagram* shows two fully independent clock domains:

- The **lpuart_pclk** clock domain

  The **lpuart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the LPUART registers are required.
- The **lpuart_ker_ck** kernel clock domain

  The **lpuart_ker_ck** is the LPUART clock source. It is independent of the **lpuart_pclk** and delivered by the RCC. So, the LPUART registers can be written/read even when the **lpuart_ker_ck** is stopped.

  When the dual clock domain feature is not supported, the **lpuart_ker_ck i**s the same as the lpuart_pclk clock.

There is no constraint between **lpuart_pclk** and **lpuart_ker_ck**: **lpuart_ker_ck** can be faster or slower than **lpuart_pclk**, with no more limitation than the ability for the software to manage the communication fast enough.

37.4.4 LPUART character description

The word length can be set to 7 or 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the LPUART_CR1 register (see *Figure 307*).

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

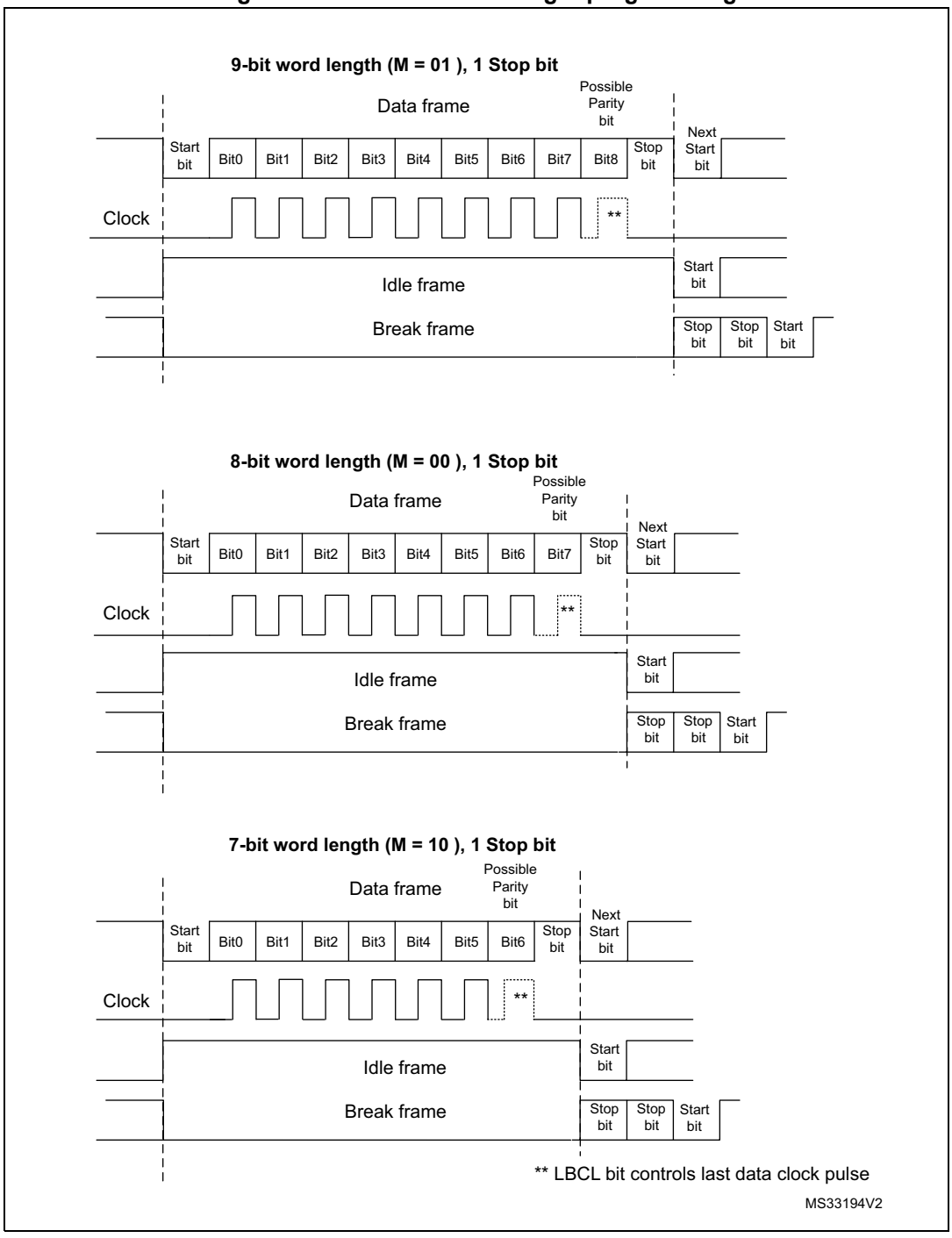
An ***idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clocks are generated when the enable bit is set for the transmitter and receiver, respectively.

The details of each block is given below.

Figure 332. LPUART word length programming



37.4.5 LPUART FIFOs and thresholds

The LPUART can operate in FIFO mode.

The LPUART comes with a transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN bit (bit 29) in the LPUART_CR1 register.

Since 9 bits the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However, the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the LPUART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through the RXFTCFG[2:0] and TXFTCFG[2:0] bitfields of the LPUART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield.

  In this case, the RXFT flag is set in the LPUART_ISR register. This means that RXFTCFG[2:0] data have been received: 1 data in LPUART_RDR and (RXFTCFG[2:0] – 1) data in the RXFIFO. As an example, when the RXFTCFG[2:0] is programmed to 101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received: FIFO size – 1 data in the RXFIFO and 1 data in the LPUART_RDR. As a result, the next received data does not set the overrun flag.
- The Tx interrupt is generated when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of the LSART_CR3 register.

37.4.6 LPUART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The transmit enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

Character transmission

During an LPUART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the LPUART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see *Section 37.4.1: LPUART block diagram*).

When FIFO mode is enabled, the data written to the LPUART_TDR register are queued in the TXFIFO.

Every character is preceded by a start bit bit, which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be 1 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the LPUART_TDR.

The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters is frozen. The current data being transmitted are lost.

An idle frame is sent after the TE bit is enabled.

Configurable stop bits

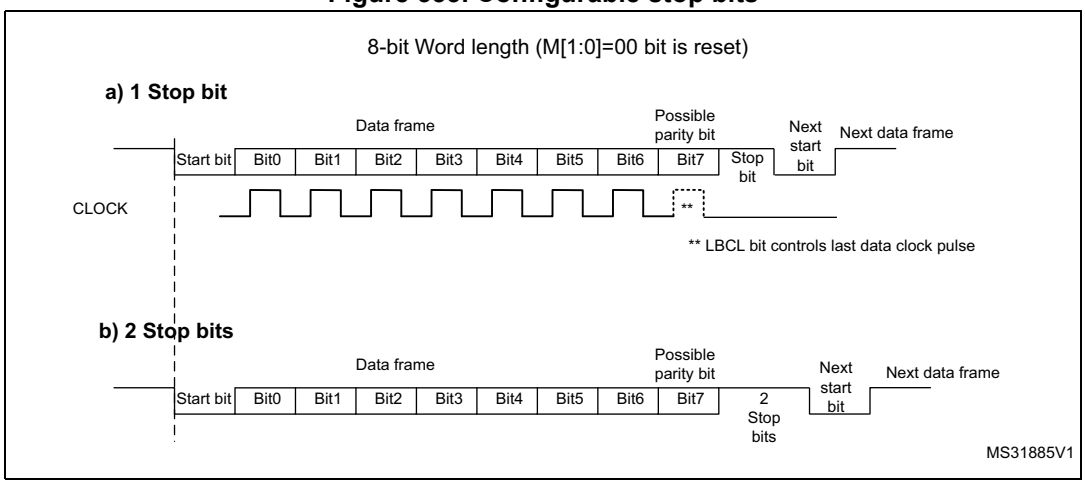
The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13, 12).

- ***1 stop bit***: This is the default value of the number of stop bits.
- ***2 stop bits***: This is supported by normal LPUART, single-wire, and modem modes.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 333. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the LPUART_BRR register.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAT) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 37.4.13: Continuous communication using DMA and LPUART*.
6. Set the TE bit in LPUART_CR1 to send an idle frame as first transmission.

7. Write the data to send in the LPUART_TDR register. Repeat this operation for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data in the LPUART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data in the LPUART_TDR adds one data to the TXFIFO. Write operations to the LPUART_TDR are performed when the TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the LPUART_TDR register, wait until TC = 1. This indicates that the transmission of the last frame has been completed.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame has been completed.
   – When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the LPUART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode disabled:

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware to indicate that:
  – the data have been moved from the LPUART_TDR register to the shift register and data transmission has started;
  – the LPUART_TDR register is empty;
  – the next data can be written to the LPUART_TDR register without overwriting the previous data.

  The TXE flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the LPUART_TDR register stores the data in the TDR register, which is copied to the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the LPUART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.
- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  – The TXFIFO is not full;
  – The LPUART_TDR register is empty;
  – The next data can be written to the LPUART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the

LPUART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

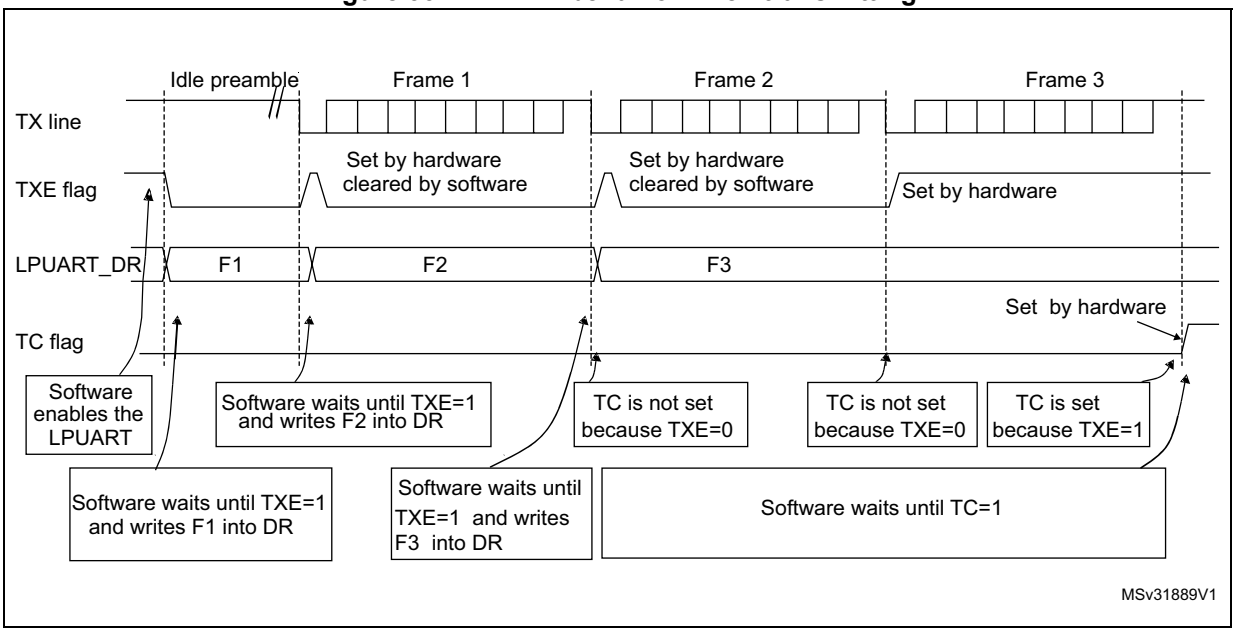
When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNEIE bit is set.

Alternatively, interrupts can be generated and data can be written to the TXFIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed threshold.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the LPUART_CR1 register.

After writing the last data in the LPUART_TDR register, it is mandatory to wait for TC = 1 before disabling the LPUART or causing the microcontroller to enter the low-power mode (see *Figure 334: TC/TXE behavior when transmitting*).

Figure 334. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see *Figure 332*).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is complete (during the stop bits after the break character). The LPUART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the LPUART to send an idle frame before the first data frame.

37.4.7 LPUART receiver

The LPUART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the LPUART_CR1 register.

Start bit detection

In the LPUART, the start bit is detected when a falling edge occurs on the Rx line, followed by a sample taken in the middle of the start bit to confirm that it is still 0. If the start sample is at 1, then the noise error flag (NE) is set, then the start bit is discarded and the receiver waits for a new start bit. Else, the receiver continues to sample all incoming bits normally.

Character reception

During an LPUART reception, data are shifted with the least significant bit first (default configuration) through the RX pin. In this mode, the LPUART_RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register LPUART_BRR
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAR) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 37.4.13: Continuous communication using DMA and LPUART*.
6. Set the RE bit LPUART_CR1. This enables the receiver, which begins searching for a start bit.

When a character is received

- When FIFO mode is disabled, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. Reading the LPUART_RDR returns the oldest data entered in the RXFIFO.

When a data is received, it is stored in the RXFIFO, together with the corresponding error bits.

- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise, or an overrun error has been detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte received and is cleared by the DMA read of the receive data register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty, that is, there is a data in the RXFIFO to be read.
- In single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read operation from the LPUART_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the LPUART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs when a character is received when RXNE has not been reset.

  Data cannot be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte received.

  An overrun error occurs if the RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – The ORE bit is set;
  – The RDR content is not lost. The previous data is available when a read to LPUART_RDR is performed.
  – The shift register is overwritten. After that, any data received during overrun is lost.
  – An interrupt is generated if either the RXNEIE bit or EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full.

  Data cannot be transferred from the shift register to the LPUART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set;
  – The first entry in the RXFIFO is not lost. It is available when a read to LPUART_RDR is performed.
  – The shift register is overwritten. After that, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE bit or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost. T

When the FIFO mode is disabled, there are two possibilities

- *If RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *If RXNE = 0, then the last valid data has already been read and there is nothing left to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.*

Selecting the clock source

The choice of the clock source is done through the clock control system (see *Section Reset and clock controller (RCC)*). The clock source must be selected through the UE bit, before enabling the LPUART.

The clock source must be selected according to two criteria:

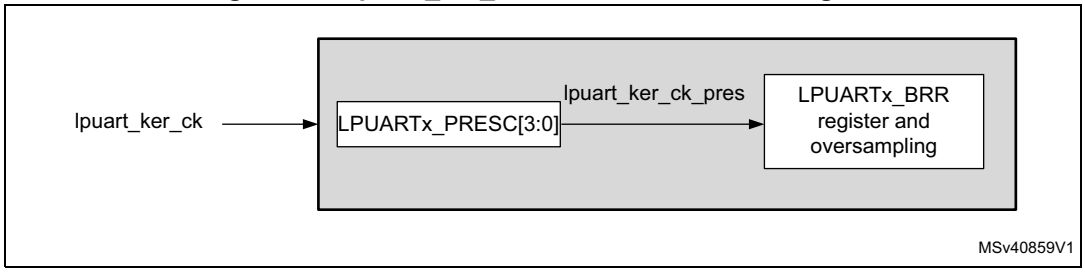
- Possible use of the LPUART in low-power mode
- Communication speed.

The clock source frequency is lpuart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the lpuart_ker_ck clock source can be configured in the RCC (see *Section Reset and clock controller (RCC)*). Otherwise, the lpuart_ker_ck is the same as lpuart_pclk.

The lpuart_ker_ck can be divided by a programmable factor in the LPUART_PRESC register.

Figure 335. lpuart_ker_ck clock divider block diagram



Some lpuart_ker_ck sources enable the LPUART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the LPUART_RDR register or by DMA.

For the other clock sources, the system must be active to enable LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming bit as close as possible to the middle of the bit-period. Only a single sample is taken of each of the incoming bits.

Note: There is no noise detection for data.

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a desynchronization or excessive noise.

When the framing error is detected:

- The FE bit is set by hardware.
- The invalid data is transferred from the Shift register to the LPUART_RDR register.
- No interrupt is generated in case of single byte communication. However, this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication, an interrupt is issued if the EIE bit is set in the LPUART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART_ICR register.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of LPUART_CR2: it can be either 1 or 2 in normal mode.

- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th, and 10th samples.
- **2 stop bits**: sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample, that is, during the second stop bit. The first stop bit is not checked for framing error.

37.4.8 LPUART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the LPUART_BRR register.

$$\text{Tx/Rx baud} = \frac{256 \times \text{lpuart_ker_ck_pres}}{\text{LPUARTDIV}}$$

LPUARTDIV is defined in the LPUART_BRR register.

Note: The baud counters are updated to the new value in the baud registers after a write operation to LPUART_BRR. Hence, the baud rate register value must not be changed during communication.

It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

lpuart_ker_ck_pres must range from 3 x baud rate to 4096 x baud rate.

The maximum baud rate that can be reached when the LPUART clock source is the LSE, is 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Table 359. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz

Baud rate		lpuart_ker_ck_pres= 32.768 kHz		
S.No	**Desired**	**Actual**	**Value programmed in the baud rate register**	**% Error = (Calculated – Desired) B.rate / Desired B.rate**
1	0.3 kbaud	300 baud	0x6D3A	0
2	0.6 kbaud	600 baud	0x369D	0
3	1200 bauds	1200.087 bauds	0x1B4E	0.007
4	2400 bauds	2400.17 bauds	0xDA7	0.007
5	4800 bauds	4801.72 bauds	0x6D3	0.035
6	9600 kbauds	9608.94 bauds	0x369	0.093

37.4.9 Tolerance of the LPUART receiver to clock deviation

The asynchronous receiver of the LPUART works correctly only if the total clock system deviation is less than the tolerance of the LPUART receiver. The causes that contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers, which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WULPUART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WULPUART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WULPUART}}{9 \times Tbit}$$

$t_{WULPUART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 360*:

- Number of stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.

Table 360. Tolerance of the LPUART receiver

M bits	768 < BRR < 1024	1024 < BRR < 2048	2048 < BRR < 4096	4096 ≤ BRR
8 bits (M = 00), 1 stop bit	1.82%	2.56%	3.90%	4.42%
9 bits (M = 01), 1 stop bit	1.69%	2.33%	2.53%	4.14%
7 bits (M = 10), 1 stop bit	2.08%	2.86%	4.35%	4.42%
8 bits (M = 00), 2 stop bits	2.08%	2.86%	4.35%	4.42%
9 bits (M = 01), 2 stop bits	1.82%	2.56%	3.90%	4.42%
7 bits (M = 10), 2 stop bits	2.34%	3.23%	4.92%	4.42%

Note: The data specified in Table 360 may slightly differ in the special case when the received frames contain some idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M = 01 or 9- bit times when M = 10).

37.4.10 LPUART multiprocessor communication

It is possible to perform LPUART multiprocessor communications (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, with its TX output connected to the RX inputs of the other LPUARTs. The others are slaves, with their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant LPUART service overhead for all nonaddressed receivers.

The nonaddressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the LPUART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two lpuart_ker_ck *cycles), otherwise mute mode might remain active.*

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in the LPUART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART_RQR register, under certain conditions.

The LPUART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the LPUART_CR1 register:

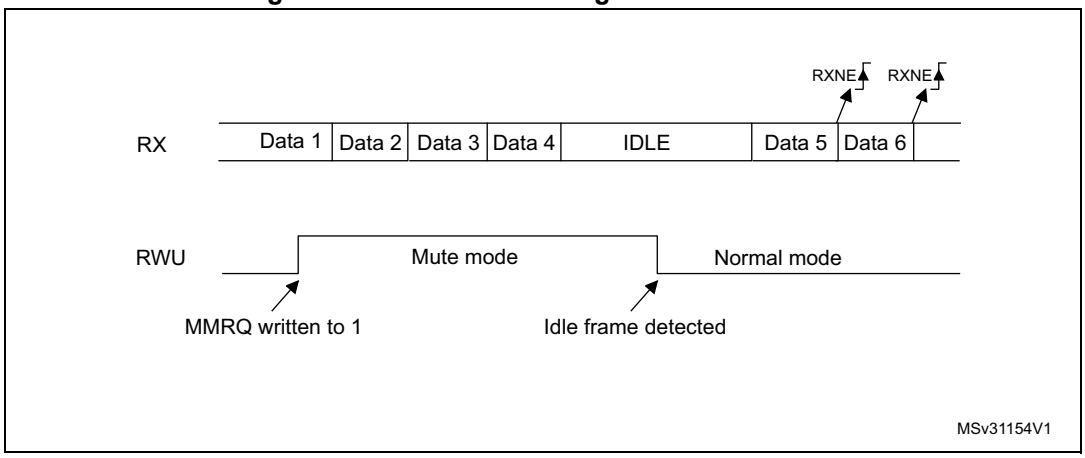
- Idle Line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE = 0)

The LPUART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The LPUART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the LPUART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 336*.

Figure 336. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the LPUART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a 1 otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7- or 4-bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address, which is programmed in the ADD bits in the LPUART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The LPUART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the LPUART enters mute mode.

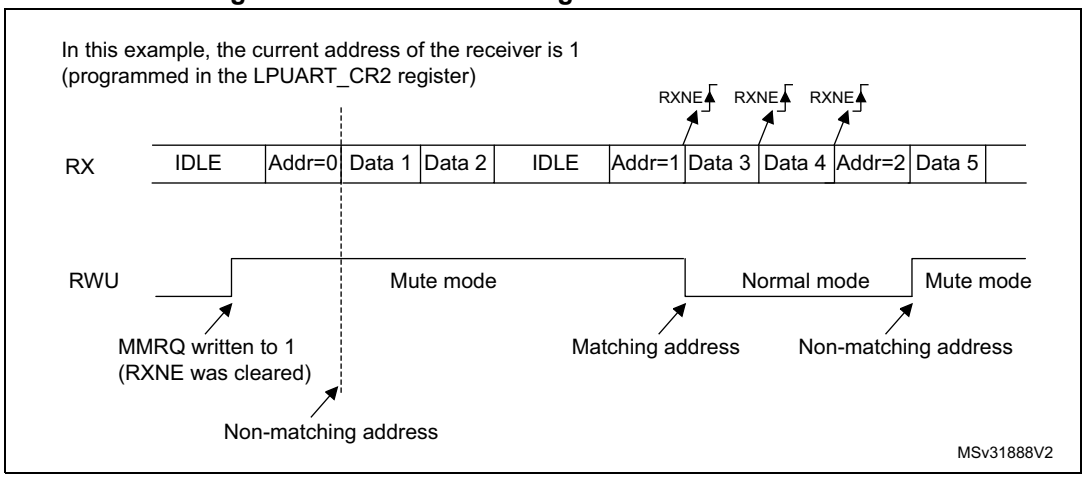
The LPUART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The LPUART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ bit is set while the receiver is sampling the last bit of a data, this data may be received before effectively entering in mute mode.

An example of mute mode behavior using address mark detection is given in *Figure 337*.

Figure 337. Mute mode using address mark detection



37.4.11 LPUART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the LPUART_CR1 register. Depending on the frame length defined by the M bits, the possible LPUART frame formats are as listed in *Table 361*.

Table 361: LPUART frame formats

M bits	PCE bit	LPUART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit.
2. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame, which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in LPUART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in LPUART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the LPUART_ISR register and an interrupt is generated if PEIE is set in the LPUART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the LPUART_ICR register.

Parity generation in transmission

If the PCE bit is set in LPUART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1)).

37.4.12 LPUART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the LPUART_CR3 register.

The LPUART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in LPUART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected.
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal LPUART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

Note: In LPUART communications, in the case of 1-stop bit configuration, the RXNE flag is set in the middle of the stop bit.

37.4.13 Continuous communication using DMA and LPUART

The LPUART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 37.3: LPUART implementation to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 37.4.7. To perform continuous communication. When FIFO is disabled, clear the TXE/ RXNE flags in the LPUART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAT bit in the LPUART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to Section *direct memory access controller*) to the LPUART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for LPUART transmission, use the following procedure (x denotes the channel number):

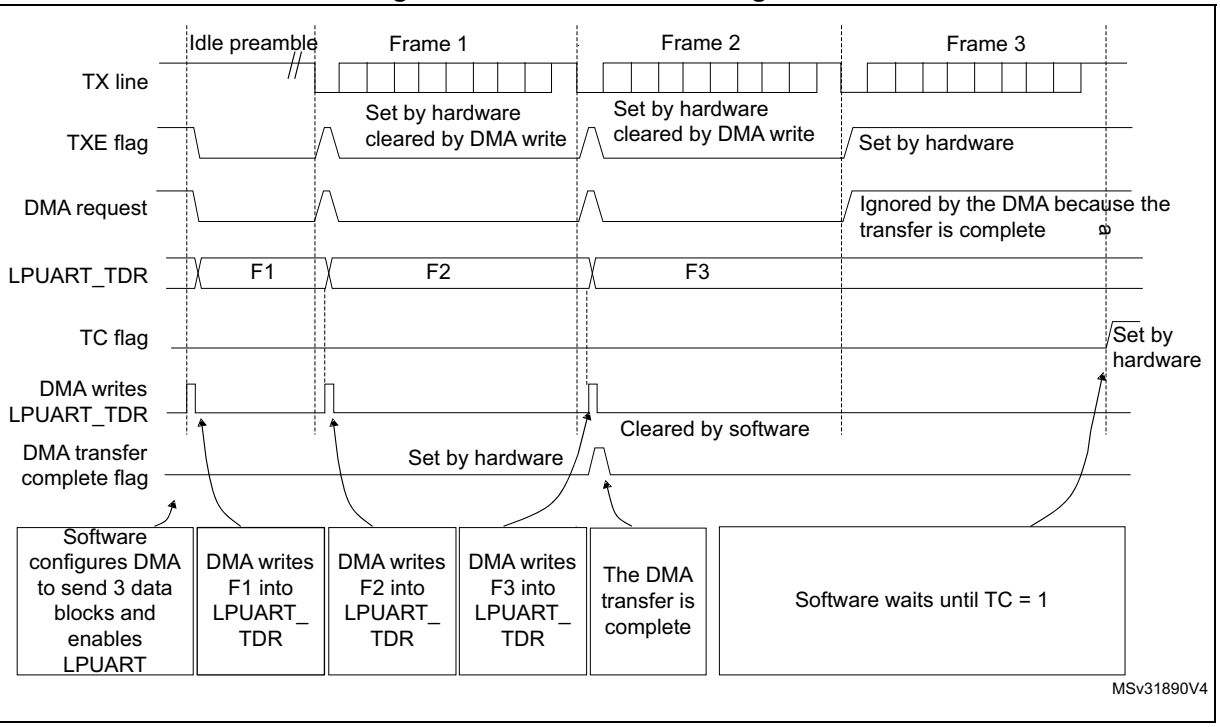
1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the LPUART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the LPUART_ISR register by setting the TCCF bit in the LPUART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (DMA transfer complete), the TC flag can be monitored to make sure that the LPUART communication has completed. This is required to avoid corrupting the last transmission before disabling the LPUART or entering low-power mode. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

Figure 338. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

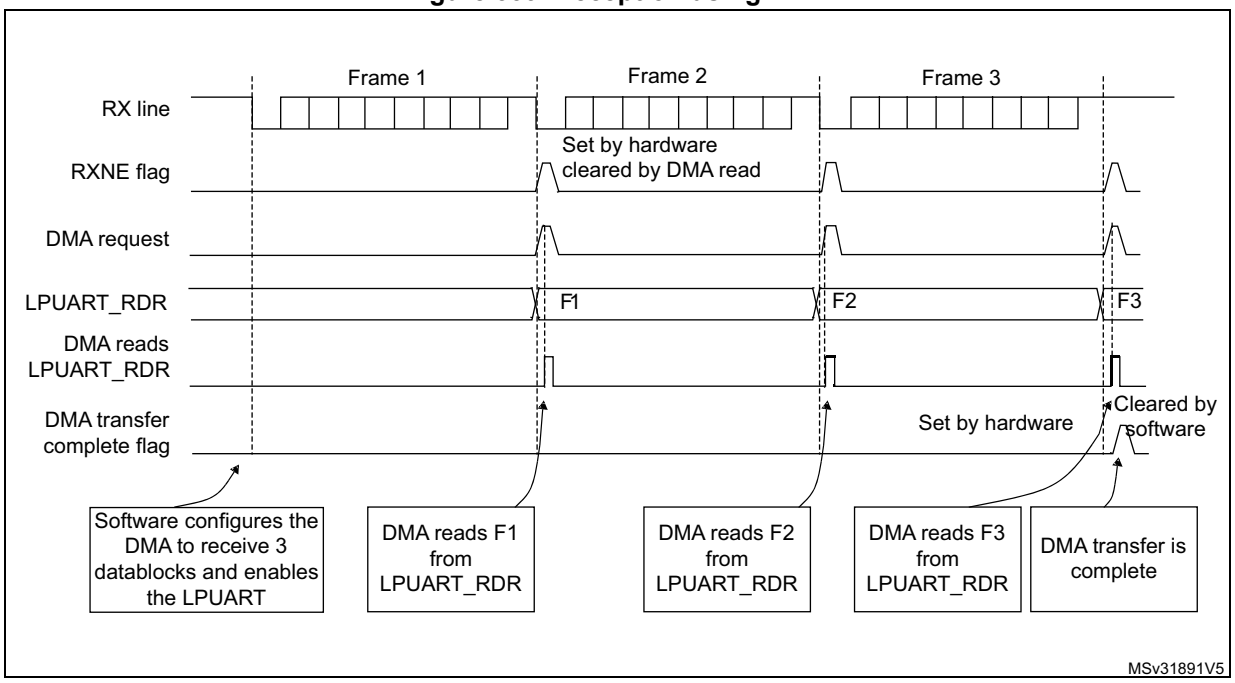
DMA mode can be enabled for reception by setting the DMAR bit in the LPUART_CR3 register. Data are loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to section *direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 339. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by receive FIFO not empty (that is, RXFNE = 1).

Error flagging and interrupt generation in multibuffer communication

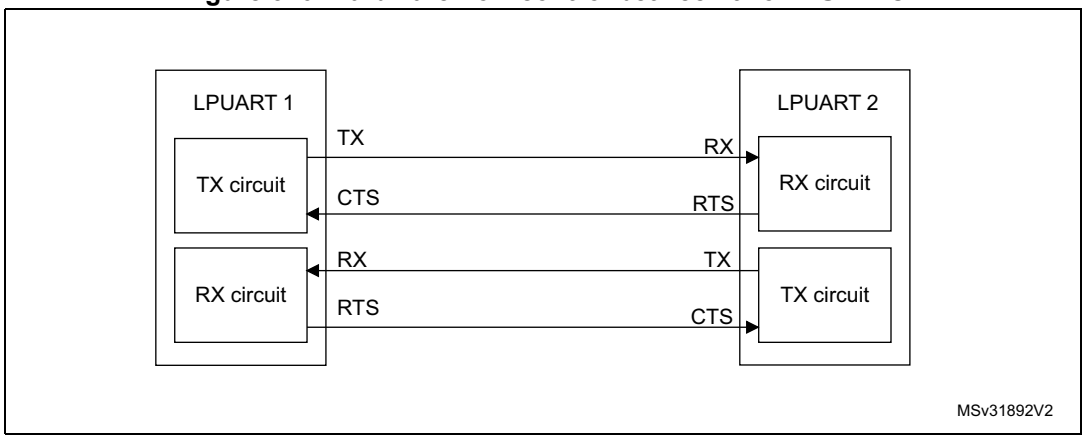
If any error occurs during a transaction In multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set.

For framing error, overrun error and noise flag, which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the LPUART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

37.4.14 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. *Figure 340* shows how to connect two devices in this mode.

Figure 340. Hardware flow control between two LPUARTs

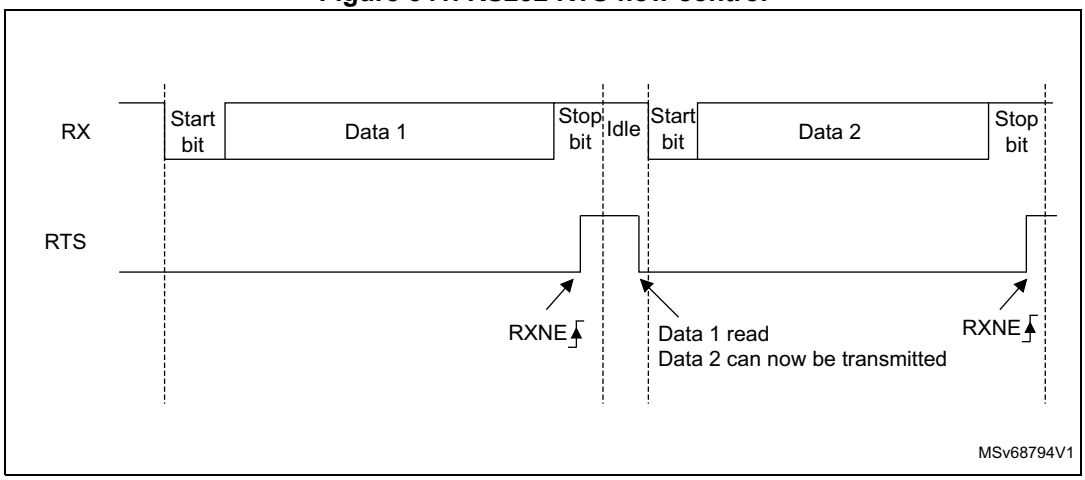


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 341* shows an example of communication with RTS flow control enabled.

Figure 341. RS232 RTS flow control



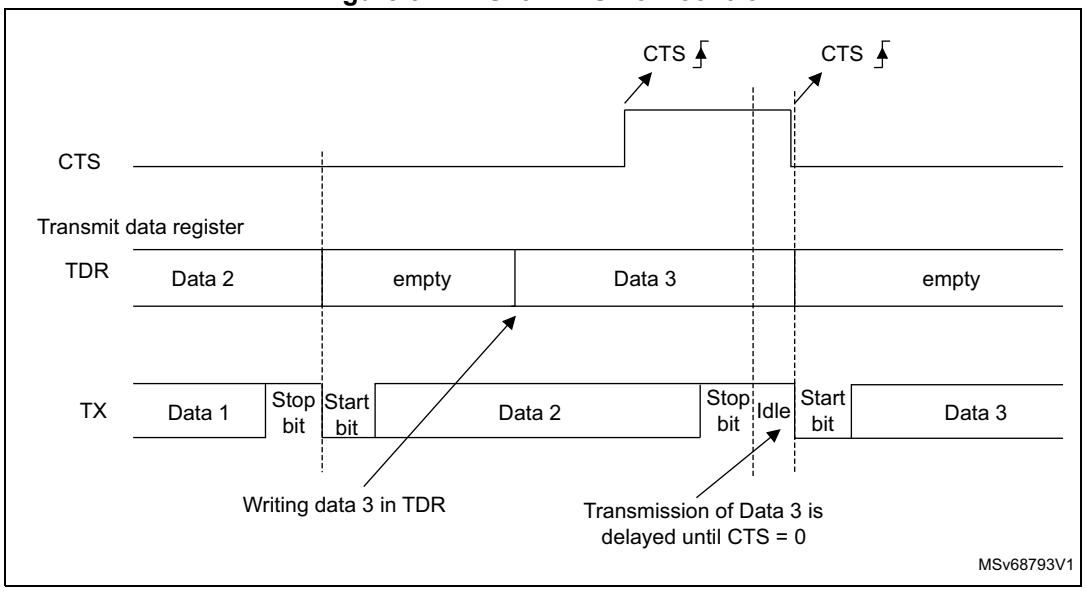
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission completes before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART_CR3 register is set. *Figure 342* shows an example of communication with CTS flow control enabled.

Figure 342. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least 3 LPUART clock source periods before the end of the current character. In addition it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the LPUART_CR3 control register. This enables activating the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the LPUART_CR1 control register. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the deactivation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the LPUART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the LPUART_CR3 control register.

The LPUART DEAT and DEDT are expressed in LPUART clock source ($f_{lpuart_ker_ck_pres}$) cycles:

- The driver enable assertion time equals
  - (1 + (DEAT x P)) x lpuart_ker_ck_pres, if P # 0
  - (1 + DEAT) x lpuart_ker_ck_pres, if P = 0
- The driver enable deassertion time equals
  - (1 + (DEDT x P)) x lpuart_ker_ck_pres, if P # 0
  - (1 + DEDT) x lpuart_ker_ck_pres, if P = 0

where P = BRR[20:11]

37.4.15 LPUART autonomous mode

The LPUART peripheral can be functional in Stop mode thanks to the autonomous mode. This mode can also be used in Run and Sleep mode. The UESM bit must be set prior to entering low-power mode.

The APB clock is requested by the peripheral each time the LPUART status needs to be updated. Once the LPUART receives the kernel and APB clocks, it generates either an interrupt or a DMA request, depending on the peripheral configuration.

If an interrupt is generated, the device wakes up from Stop mode. If no interrupt is generated, the device remains in Stop mode but the APB clock is still available for the LPUART and all the autonomous peripherals enabled in the reset and clock controller (RCC). If DMA requests are enabled, the data are directly transferred to/from the SRAM thanks to the DMA while the product remains in Stop mode.

LPUART transmission mode

In transmission, the APB clock is requested only when the TE bit is set and in the following cases:

- If the FIFO mode is enabled, the APB clock is requested when
  - The TxFIFO is empty (TXFE = 1) and the corresponding interrupt is enabled (TXFEIE = 1)
  - The TxFIFO threshold is reached (TXFT = 1) and the corresponding interrupt is enabled (TXFTIE = 1)
  - The TxFIFO is not full (TXFNF = 1) and the corresponding interrupt or DMA is enabled (TXFNFIE = 1 or DMAT = 1)
- If the FIFO mode is disabled, the APB clock is requested as soon as data are transferred to the shift register. The DMA or associated interrupt must be enabled.

The TE bit is set by hardware if an asynchronous trigger is detected.

A transmission is automatically launched when an asynchronous trigger is detected in Run, Sleep, or Stop mode. The trigger is selected through the TRIGSEL bit in the LPUART_AUTOCR register. It sets the TE bit in the LPUART_CR1 register and generates an APB clock request to enable the transfer. The APB clock is requested until the transmission completes and the TE bit is cleared by hardware when the programmed number of data to be transmitted (TDN bitfield in the LPUART_AUTOCR register) is reached. In this case, the TC flag is set when the number of data to be transmitted is reached and the last byte is transmitted.

LPUART reception mode

- If the FIFO mode is enabled, the APB clock is requested when
  - The RxFIFIO is full (RXFF = 1) and the corresponding interrupt is enabled (RXFFIE = 1)
  - The RxFIFO threshold is reached (RXFT = 1) and the corresponding interrupt is enabled (RXFTIE = 1)
  - The RxFIFO is not empty (RXFNE = 1) and the corresponding interrupt or DMA is enabled (RXFNEIE= 1)
- If the FIFO mode is disabled, the APB clock is requested when the LPUART finishes sampling data and it is ready to be written in the LPUART_RDR. The DMA or the associated interrupt must be enabled.

Note: The APB clock is requested in reception mode when an overrun error occurs (ORE = 1). The EIE bit must be set to enable the generation an interrupt and waking up the MCU, and the OVRDIS bit must remain cleared. The APB clock request is kept until the interrupt flag is cleared.

In reception mode, the APB clock is requested when a Parity/Noise/Framing error occurs and the DMA is used for reception. The APB clock request is kept until the interrupt flag is cleared.

Determining the maximum LPUART baud rate that enables to correctly wake up the MCU from low-power mode

The maximum baud rate that enables to correctly wake up the MCU from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the LPUART receiver tolerance (see *Section 37.4.9: Tolerance of the LPUART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = 01, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 360: Tolerance of the LPUART receiver*, the LPUART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < LPUART receiver tolerance

$D_{WUmax} = t_{WULPUART} / (11 \times T_{bit\ Min})$

$T_{bit\ Min} = t_{WULPUART} / (11 \times D_{WUmax})$

where $t_{WULPUART}$ is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC, and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In fact, we need to consider at least the lpuart_ker_ck inaccuracy.

For example, if HSI is used as lpuart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WULPUART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = 3.41% – 1% = 2.41%

$T_{bit\ min}$ = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate that enables to wake up correctly from low-power mode is: 1/11.32 µs = 88.36 kbauds.

37.5 LPUART in low-power modes

Table 362. Effect of low-power modes on the LPUART

Mode	Description
Sleep	No effect. LPUART interrupts cause the device to exit Sleep mode.
Stop 0 and Stop 1(1)	The content of the LPUART registers is kept. If the LPUART is clocked by an oscillator available in Stop mode, transfers in asynchronous mode are functional. DMA requests are functional, and the interrupts cause the device to exit Stop mode.
Stop 2	The content of the LPUART registers is kept. If the LPUART is clocked by an oscillator available in Stop mode, transfers in asynchronous mode are functional. The interrupts cause the device to exit Stop 2 mode.
Stop 3	The content of the LPUART registers is kept. No interrupt is generated.
Standby	The LPUART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 37.3: LPUART implementation* to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

37.6 LPUART interrupts

Refer to *Table 363* for a detailed description of all LPUART interrupt requests.

Table 363. LPUART interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop (1) modes	Exit from Standby mode
LPUART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	Yes	No
	Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFIFO full		Yes	
	Transmit FIFO Empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission Complete	TC	TCIE	Write TDR or write 1 in TCCF		Yes	
	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	
	Receive FIFO Not Empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO Full	RXFF(2)	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RX-NEIE/RX FNEIE	Write 1 in ORECF		Yes	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		Yes(3)	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		Yes(3)	
	Overrun error in multibuffer communication.	ORE(4)		Write 1 in ORECF		Yes	
	Framing Error in multibuffer communication.	FE		Write 1 in FECF		Yes(3)	
	Character match	CMF	CMIE	Write 1 in CMCF		Yes(5)	

1. The LPUART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to *Section 37.3: LPUART implementation* for the list of supported Stop modes.

2. RXFF flag is asserted if the LPUART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in LPUART_RDR. In Stop mode, LPUART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
3. Parity/Noise/Framing error interrupts enable waking up from Stop modes when the DMA is used.
4. When OVRDIS = 0.
5. The DMA must be used when the FIFO mode is enabled.

37.7 LPUART registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

37.7.1 LPUART control register 1 (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE:**RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when RXFF = 1 in the LPUART_ISR register

Bit 30 **TXFEIE:**TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when TXFE = 1 in the LPUART_ISR register

Bit 29 **FIFOEN:**FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 Start bit, 9 Data bits, n stop bits

M[1:0] = ‘10: 1 Start bit, 7 Data bits, n stop bits

This bit can only be written when the LPUART is disabled (UE = 0).

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 36.5.21: RS232 hardware flow control and RS485 driver enable*.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal.It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 37.4.14: RS232 hardware flow control and RS485 driver enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXFNF = 1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXFNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. When the autonomous mode is not used, TE bit is set and cleared by software. When the autonomous mode is used, TE bit becomes a status bit, which is set and cleared by hardware.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the LPUART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register. In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in low-power mode

When this bit is cleared, the LPUART cannot request its kernel clock and is not functional in low-power mode.

When this bit is set, the LPUART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: LPUART not functional in low-power mode.

1: LPUART functional in low-power mode.

Note: The UESM bit must be set at the initialization phase.

If the LPUART *does not support the wake-up from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 37.3: LPUART implementation.*

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

37.7.2 LPUART control register 1 [alternate] (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
		rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN:**FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 Start bit, 9 Data bits, n stop bits

M[1:0] = ‘10: 1 Start bit, 7 Data bits, n stop bits

This bit can only be written when the LPUART is disabled (UE = 0).

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 36.5.21: RS232 hardware flow control and RS485 driver enable*.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal.It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 37.4.14: RS232 hardware flow control and RS485 driver enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXE = 1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. When the autonomous mode is disabled, TE bit is set and cleared by software. When the autonomous mode is enabled, TE bit becomes a status bit, which is set and cleared by hardware.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the LPUART acts as a transmitted, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register. In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in low-power mode

When this bit is cleared, the LPUART cannot request its kernel clock and is not functional in low-power mode.

When this bit is set, the LPUART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: LPUART not functional in low-power mode.

1: LPUART functional in low-power mode.

Note: The UESM bit must be set at the initialization phase.

If the LPUART does not support the wake-up from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 37.3: LPUART implementation.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

37.7.3 LPUART control register 2 (LPUART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	Res.	STOP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
rw		rw	rw								rw				

Bits 31:24 **ADD[7:0]**: Address of the LPUART node

These bits give the address of the LPUART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter must be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When a character, received during low-power mode, corresponds to the character programmed through ADD[7:0] bitfield, the CMF flag is set and wakes up the device from low-power mode if the corresponding interrupt is enabled by setting CMIE bit.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the LPUART is disabled (UE = 0).

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.
0: data is transmitted/received with data bit 0 first, following the start bit.
1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.
0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.
0: TX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)
1: TX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).
This enables the use of an external inverter on the TX line.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.
0: RX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0/mark)
1: RX pin signal values are inverted. ((V_{DD} = 0/mark, Gnd = 1 / idle).
This enables the use of an external inverter on the RX line.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.
0: TX/RX pins are used as defined in standard pinout
1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **STOP[1:0]**: Stop bits
These bits are used for programming the stop bits.
00: 1 stop bit
01: Reserved.
10: 2 stop bits
11: Reserved
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **ADDM7**:7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the LPUART is disabled (UE = 0)
Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0 Reserved, must be kept at reset value.

37.7.4 LPUART control register 3 (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			Res.	TXFTIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw		rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth.
001:TXFIFO reaches 1/4 of its depth.
110:TXFIFO reaches 1/2 of its depth.
011:TXFIFO reaches 3/4 of its depth.
100:TXFIFO reaches 7/8 of its depth.
101:TXFIFO becomes empty.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG[2:0].

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth.
001:Receive FIFO reaches 1/4 of its depth.
110:Receive FIFO reaches 1/2 of its depth.
011:Receive FIFO reaches 3/4 of its depth.
100:Receive FIFO reaches 7/8 of its depth.
101:Receive FIFO becomes full.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 24 Reserved, must be kept at reset value.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG[2:0].

Bits 22:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 13 **DDRE**: DMA Disable on reception Error
0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: This control bit enables *checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF = 1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the LPUART is disabled (UE = 0)

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half-duplex mode is not selected

1: Half-duplex mode is selected

This bit can only be written when the LPUART is disabled (UE = 0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register).

0: Interrupt is inhibited

1: An interrupt is generated when FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register.

37.7.5 LPUART control register 3 [alternate] (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 13 **DDRE**: DMA Disable on reception Error
0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: This control bit enables *checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF = 1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable
0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.
This bit can only be written when the LPUART is disabled (UE = 0)

Bit 8 **RTSE**: RTS enable
0: RTS hardware flow control disabled
1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection
Selection of single-wire half-duplex mode
0: Half-duplex mode is not selected
1: Half-duplex mode is selected
This bit can only be written when the LPUART is disabled (UE = 0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register.

37.7.6 LPUART baud rate register (LPUART_BRR)

This register can only be written when the LPUART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **BRR[19:0]:** LPUART baud rate division (LPUARTDIV)

Note: It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

Provided that LPUART_BRR must be ≥ 0x300 and LPUART_BRR is 20 bits, a care must be taken when generating high baud rates using high lpuart_ker_ck_pres values. lpuart_ker_ck_pres must be in the range [3 x baud rate..4096 x baud rate].

37.7.7 LPUART request register (LPUART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
											w	w	w	w	

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This sets the flag TXFE (TXFIFO empty, bit 23 in the LPUART_ISR register).

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit clears the RXNE flag.

This enables discarding the received data without reading it, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the LPUART in Mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: If the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 Reserved, must be kept at reset value.

37.7.8 LPUART interrupt and status register (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0080 00C0

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	Res.	RXFF	TXFE	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY
				r	r		r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of LPUART_CR3 register.An interrupt is generated if the TXFTIE bit = 1 (bit 31) in the LPUART_CR3 register.

0: TXFIFO does not reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield of the LPUART_CR3 register, that is, the Receive FIFO contains RXFTCFG[2:0] data. An interrupt is generated if the RXFTIE bit = 1 (bit 28) in the LPUART_CR3 register.

0: Receive FIFO does not reach the programmed threshold.

1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG[2:0] threshold is configured to 101, the RXFT flag is set if RXFIFO size data are available, that is, (RXFIFO size – 1) data in the RXFIFO and 1 data in the LPUART_RDR. Consequently, the (RXFIFO size + 1) th received data does not cause an overrun error. The overrun error occurs after receiving the (RXFIFO size + 2) th data.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the LPUART_RDR register.

An interrupt is generated if the RXFFIE bit = 1 in the LPUART_CR1 register.

0: RXFIFO is not Full.

1: RXFIFO is Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the LPUART_RQR register.

An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the LPUART_CR1 register.

0: TXFIFO is not empty.

1: TXFIFO is empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE = 1in the LPUART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: reception ongoing

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART_TDR. Every write in the LPUART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART_TDR.

The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).

An interrupt is generated if the TXFNFIE bit = 1 in the LPUART_CR1 register.

0: Data register is full/Transmit FIFO is full.

1: Data register/Transmit FIFO is not full.

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register.

The TC flag behaves as follows:

- When TDN = 0, the TC flag is set when the transmission of a frame containing data has completed and when TXE/TXFE is set.
- When TDN is equal to the number of data in the TXFIFO, the TC flag is set when TXFIFO is empty and TDN is reached.
- When TDN is greater than the number of data in the TXFIFO, TC remains cleared until the TXFIFO is filled again to reach the programmed number of data to be transferred.
- When TDN is less than the number of data in the TXFIFO, TC is set when TDN is reached even if the TXFIFO is not empty.

An interrupt is generated if TCIE = 1 in the LPUART_CR1 register.

The TC bit is cleared by software, by writing 1 to the TCCF of the LPUART_ICR register, or by writing to the LPUART_TDR register.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the LPUART_RDR register. Every read of the LPUART_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXFNEIE = 1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXFNEIE = 1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

This error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.

When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the LPUART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.

0: No parity error

1: Parity error

Note: This error is associated with the character in the LPUART_RDR.

37.7.9 LPUART interrupt and status register [alternate] (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY
									r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE = 1in the LPUART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: Reception ongoing

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the LPUART_TDR register has been transferred into the shift register. It is cleared by a write to the LPUART_TDR register.

An interrupt is generated if the TXEIE bit = 1 in the LPUART_CR1 register.

0: Data register full

1: Data register empty

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.

An interrupt is generated if TCIE = 1 in the LPUART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the LPUART_ICR register or by writing to the LPUART_TDR register.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the LPUART_RDR shift register has been transferred to the LPUART_RDR register. It is cleared by a read to the LPUART_RDR register. The

RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXNEIE = 1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXNE = 1 (RXFF = 1 in case FIFO mode is enabled). It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXNEIE = 1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE/RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the LPUART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.
An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.
0: No parity error
1: Parity error

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

37.7.10 LPUART interrupt flag clear register (LPUART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMCF	Res.
														w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
						w			w		w	w	w	w	w

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the LPUART_ISR register.

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag

Writing 1 to this bit clears the CTSIF flag in the LPUART_ISR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag

Writing 1 to this bit clears the TC flag in the LPUART_ISR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IDLECF**: Idle line detected clear flag

Writing 1 to this bit clears the IDLE flag in the LPUART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag

Writing 1 to this bit clears the ORE flag in the LPUART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the LPUART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the LPUART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the LPUART_ISR register.

37.7.11 LPUART receive data register (LPUART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see *Section 37.4.1: LPUART block diagram*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

37.7.12 LPUART transmit data register (LPUART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see *Section 37.4.1: LPUART block diagram*).

When transmitting with the parity enabled (PCE bit set to 1 in the LPUART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF = 1.

37.7.13 LPUART prescaler register (LPUART_PRESC)

This register can only be written when the LPUART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The LPUART input clock can be divided by a prescaler:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Others: Reserved, must not be used.

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011, that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 37.3: LPUART implementation.

37.7.14 LPUART autonomous mode control register (LPUART_AUTOCR)

Address offset: 0x30

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGSEL[3:0]				IDLEDIS	TRIGEN	TRIGPOL
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:19 **TRIGSEL[3:0]**: Trigger selection bits

Refer to *Description LPUART interconnections*.

This bitfield can be written only when the UE bit is cleared in LPUART_CR1 register.

0000: lpuart_trg0 selected

0001: lpuart_trg1 selected

...

1111: lpuart_trg15 selected

Note: This bitfield can be written only when the UE bit of LPUART_CR1 register is cleared.

Bit 18 **IDLEDIS**: Idle frame transmission disable bit after enabling the transmitter

0: Idle frame sent after enabling the transmitter (TE = 1 in LPUART_CR1)

1: Idle frame not sent after enabling the transmitter

Note: This bitfield can be written only when the UE bit of LPUART_CR1 register is cleared.

Bit 17 **TRIGEN**: Trigger enable bit

0: Trigger disabled

1: Trigger enabled

Note: This bitfield can be written only when the UE bit of LPUART_CR1 register is cleared.

When a trigger is detected, TE is set to 1 in LPUART_CR1 and the data transfer is launched.

Bit 16 **TRIGPOL**: Trigger polarity bit

This bitfield can be written only when the UE bit is cleared in LPUART_CR1 register.

0: Trigger active on rising edge

1: Trigger active on falling edge

Bits 15:0 **TDN[15:0]**: TDC transmission data number

This bitfield enables the programming of the number of data to be transmitted. It can be written only when UE is cleared in LPUART_CR1.

37.7.15 LPUART register map

Table 364. LPUART register map and reset values

Offset	Register name	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
0x00	LPUART_CR1FIFO mode enabled	RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]					Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 364. LPUART register map and reset values (continued)

Offset	Register name	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
0x00	LPUART_CR1FIFO mode disabled	Res.	Res.	FIFOEN	M1	Res.	Res.	DEAT[4:0]						DEDT[4:0]				Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value			0	0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	LPUART_CR2	ADD[7:0]								Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	Res.	STOP [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0		0	0								0				
0x08	LPUART_CR3 FIFO mode enabled	TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			Res.	TXFTIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	Res.	EIE
	Reset value	0	0	0	0	0	0	0		0								0	0	0	0		0	0	0	0	0			0				0
0x08	LPUART_CR3 FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	Res.	EIE
	Reset value																	0	0	0	0		0	0	0	0	0			0				0
0x0C	LPUART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10-0x14	Reserved																																
0x18	LPUART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
	Reset value																												0	0	0	0	
0x1C	LPUART_ISRFIFO mode enabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFF	TXFF	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value								0	1	0	0		0	0	0	0						0	0		1	0	0	0	0	0	0	0
0x1C	LPUART_ISRFIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value										0	0		0	0	0	0						0	0		1	0	0	0	0	0	0	0
0x20	LPUART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
	Reset value															0								0			0		0	0	0	0	0
0x24	LPUART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x28	LPUART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x2C	LPUART_PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0

Table 364. LPUART register map and reset values (continued)

Offset	Register name	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
0x30	LPUART_ AUTOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGSEL [3:0]				IDLEDIS	TRIGEN	TRIGPOL	TDN[15:0]																
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

38 Serial peripheral interface (SPI)

38.1 SPI introduction

The serial peripheral interface (SPI) can be used to communicate with external devices while using the specific synchronous protocol. The SPI protocol supports half-duplex, full-duplex, and simplex synchronous, serial communication with external devices. The interface can be configured as master or slave and can operate in multislave or multimaster configurations. The device configured as master provides a communication clock (SCK) to the slave device. The slave select (NSS) and ready (RDY) signals can be applied optionally just to set up communication with a concrete slave and to ensure it handles the data flow properly. The Motorola data format is used by default, but some other specific modes are supported as well.

38.2 SPI main features

- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- 8-bit or 16-bit data size selection
- Multimaster or multislave mode capability
- Dual clock domain, the peripheral kernel clock is independent from the APB bus clock
- Baud rate prescaler up to kernel frequency/2 or bypass from RCC in master mode
- Protection of configuration and setting
- Slave select hardware or software management for both master and slave
- Adjustable minimum delays between data and between NSS and data flow
- Configurable NSS signal polarity and timing, MISO x MOSI swap capability
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Programmable number of data within a transfer to control NSS and CRC
- Dedicated transmission and reception flags with interrupt capability
- SPI Motorola and TI format support
- Hardware CRC can verify the integrity of the communication at the end of a transfer by:
  - Adding CRC value in Tx mode
  - Automatic CRC error checking for Rx mode
- Error detection with interrupt capability in case of data overrun, CRC error, data underrun, the mode fault, and frame error, depending on the operating mode
- Two 8-bit width embedded Rx and Tx FIFOs (FIFO size depends on instances)
- Configurable FIFO thresholds (to handle the data packets)
- Capability to handle data streams by system DMA controller
- Configurable behavior at slave underrun condition (support of cascaded circular buffers)

- Autonomous functionality in Stop modes (handling of the transfer flow and required clock distribution) with wake-up from Stop capability
- Optional status pin RDY signalizing that the slave device is ready to handle the data flow

38.3 SPI implementation

The table below describes the SPI implementation.

Table 365. SPI features

SPI feature	SPI3 (limited-featured instance)
Data size	8/16 bits
CRC computation	CRC polynomial length configurable from 9 to 17 bits
FIFO size	8x8 bits
FIFO threshold	1 to 4 data
Number of data control	Up to 1023, no data counter
I2S feature	No
Autonomous in Stop modes with wake-up capability	Yes
Autonomous in Stop 0, Stop 1, and Stop 2 modes with wake-up capability	Yes

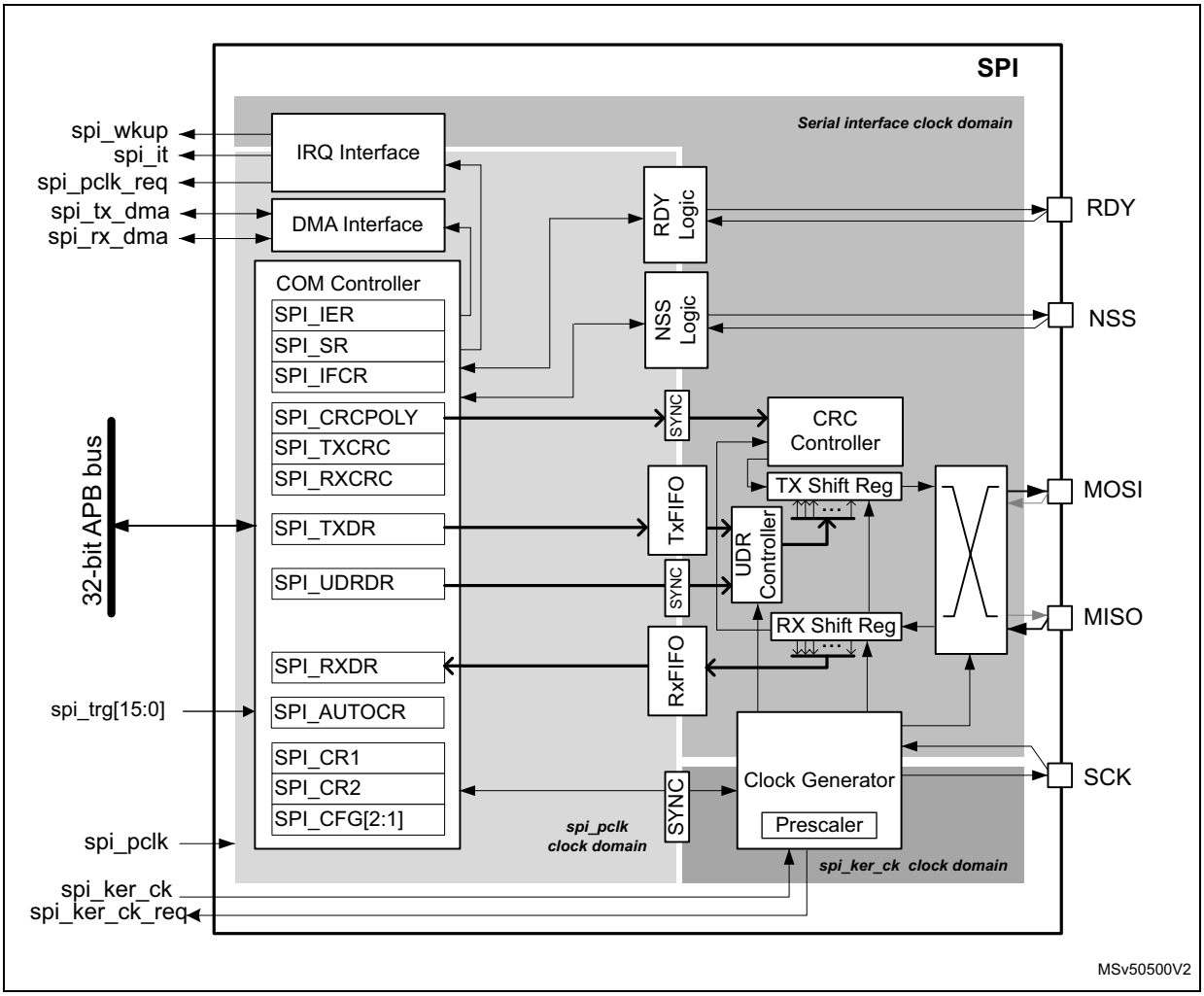
Note: For detailed information about instance capabilities to exit from Stop and Standby modes, refer to Table 370: SPI wake-up and interrupt requests and Table 95: Functionalities depending on the working mode.

38.4 SPI functional description

38.4.1 SPI block diagram

The SPI enables synchronous, serial communications between the MCU and external devices. The application software can manage the communication by polling the status flag or using a dedicated SPI interrupt. The main SPI elements and their interactions are shown in *Figure 343*.

Figure 343. SPI block diagram



The simplified scheme of *Figure 343* shows three fully independent clock domains:

- The **spi_pclk** clock domain
- The **spi_ker_ck** kernel clock domain
- The serial interface clock domain

All the control and status signals between these domains are strictly synchronized. There is no specific constraint concerning the frequency ratio between these clock signals. The user has to consider a ratio compatible with the data flow speed to avoid data underrun or overrun events.

The **spi_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the SPI registers are required.

The SPI working in slave mode handles a data flow using the serial interface clock derived from the external SCK signal provided by the external master SPI device. That is why the SPI slave is able to receive and send data even when the **spi_pclk** and **spi_ker_ck** clock signals are inactive. As a consequence, a specific slave logic working within the serial interface clock domain needs some additional traffic to be set up correctly (for example when underrun or overrun is evaluated, see *Section 38.5.2* for details). This cannot be done when the bus becomes idle. In some specific cases, the slave even requires the clock generator working (see *Section 38.5.1*).

When the SPI works as a master, the RCC must provide the **spi_ker_ck** kernel clock to the peripheral during communication to feed the serial interface clock via the clock generator where it can be divided by prescaler or bypassed. The signal is then provided to the slaves via the SCK pin and internally to the serial interface domain of the master.

38.4.2 SPI pins and internal signals

Up to five I/O pins are dedicated to SPI communication with external devices.

- **MISO:** master in / slave out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** master out / slave in data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** serial clock output pin for SPI masters and input pin for SPI slaves.
- **NSS:** slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - Select an individual slave device for communication
  - Synchronize the data frame, or
  - Detect a conflict between multiple masters

  See *Section 38.4.7* for details.
- **RDY:** optional status pin signaling slave FIFO occupancies and so the slave availability to continue the transfer without any risk of data flow corruption. It can be checked by the master to control the temporal suspension of the ongoing communication.

The SPI bus enables the communication between one master device and one or more slave devices. The bus consists of at least two wires: one for the clock signal and the other for synchronous data transfer. Other signals are optional and can be added depending on the data exchange between SPI nodes and their communication control management.

Refer to *Table 366* and *Table 367* for the list of SPI input / output pins and internal signals.

Table 366. SPI input/output pins

Pin name	I/O type	Description
MISO(1)	Input/output	Master data input / slave data output
MOSI(1)	Input/output	Master data output / slave data input
SCK	Input/output	Master clock output / slave clock input
NSS	Input/output	Master output / slave selection input
RDY	Input/output	SPI master input / slave FIFOs status occupancy output

1. Functionality of MOSI and MISO pins can be swapped. Their directions may vary in SPI bidirectional half-duplex mode.

Description of SPI input/output signals

Table 367. SPI internal input/output signals

Signal name	Signal type	Description
spi_pclk	Input	SPI clock signal feeds the peripheral bus interface
spi_ker_ck	Input	SPI kernel clock
spi_ker_ck_req	Output	SPI kernel clock request
spi_pclk_req	Output	SPI clock request
spi_wkup	Output	SPI provides a wake-up interrupt
spi_it	Output	SPI global interrupt
spi_tx_dma	Input/output	SPI transmit DMA request
spi_rx_dma	Input/output	SPI receive DMA request
spi_trg[15:0]	Input	SPI triggers

Description of SPI interconnections

Table 368. SPI interconnection (SPI3)

Signal name	Trigger source
spi_itr0	lpdma1_ch0_tcf
spi_itr1	lpdma1_ch1_tcf
spi_itr2	lpdma1_ch2_tcf
spi_itr3	lpdma1_ch3_tcf
spi_itr4	exti4
spi_itr5	exti8
spi_itr6	lptim1_ch1
spi_itr7	-
spi_itr8	-
spi_itr9	-
spi_itr10	rtc_alra_trg
spi_trg11	rtc_wut_trg
spi_trg12	-
spi_trg13	-
spi_trg14	-

38.4.3 SPI communication general aspects

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use two or three wires (with software slave select management) or three or four wires (with hardware slave select management). The communication is always initiated and controlled by the master. The master provides a clock signal on the SCK line and selects or synchronizes slaves for communication by NSS line when it is managed by hardware.

The data between the master and the slave flow synchronously on the MOSI and/or MISO lines.

38.4.4 Communications between one master and one slave

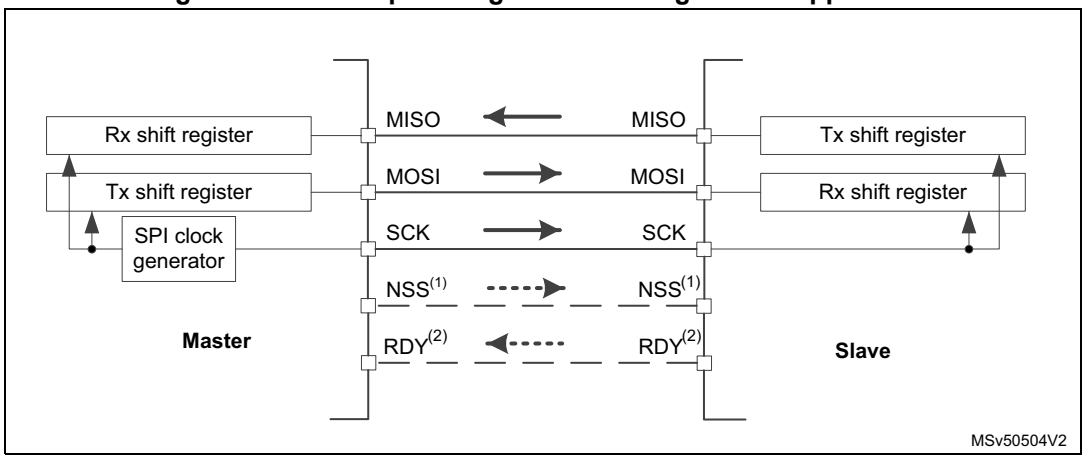
The communication flow can use one of three possible modes: the full-duplex (three wires) mode, half-duplex (two wires) mode, or the simplex (two wires) mode. The NSS signal is optional in single master-slave configuration and is often not connected between the two communication nodes. Nevertheless, the NSS signal can be helpful in this configuration to synchronize the data flow and it is used by default for some specific SPI modes (for example the TI mode).

The next optional RDY signal can help to ensure the correct management of all the transferred data at slave side.

Full-duplex communication

By default, the SPI is configured for full-duplex communication (bits COMM[1:0] = 00 in the SPI_CFG2 register). In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During the SPI communication, the data are shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line simultaneously. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 344. Full-duplex single master/ single slave application

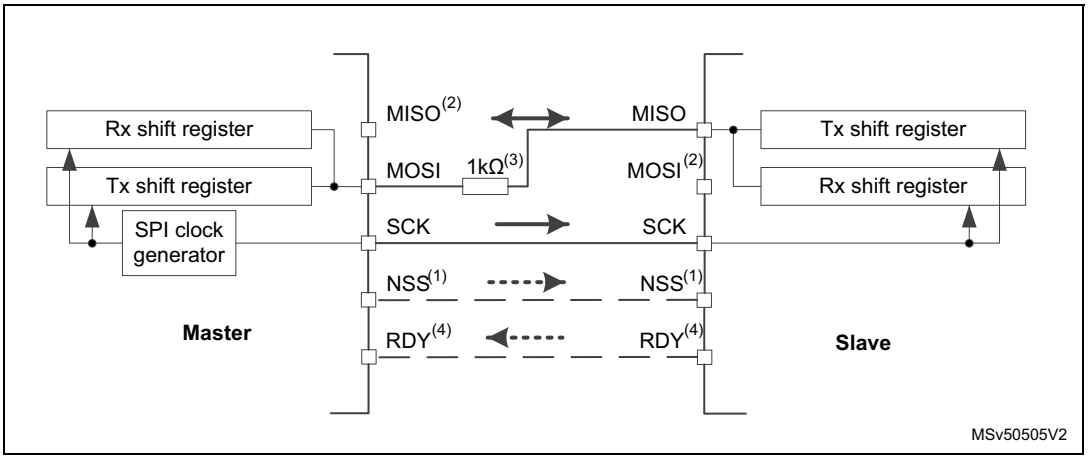


1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 38.4.7*).
2. The RDY signal provided by the slave can be read by the master optionally.

Half-duplex communication

The SPI can communicate in half-duplex mode by setting COMM[1:0] = 11 in the SPI_CFG2 register. In this configuration, one single cross-connection line is used to link the shift registers of the master and slave together. During this communication, the data are synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the HDDIR bit in their SPI_CR1 registers. Note that the SPI must be disabled when changing the direction of the communication. In this configuration, the MISO pin at master and the MOSI pin at slave are free for other application uses and act as GPIOs.

Figure 345. Half-duplex single master/ single slave application



1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 38.4.7*).
2. In this configuration, the MISO pin at master and MOSI pin at slave can be used as GPIOs
3. A critical situation can happen when the communication direction is not changed synchronously between two nodes working in bidirectional mode. The new transmitter accesses the common data line while the former transmitter still keeps an opposite value on the line (the value depends on the SPI configuration and communicated data). The nodes can conflict temporarily with opposite output levels on the line until the former transmitter changes its data direction setting. It is suggested to insert a serial resistance between MISO and MOSI pins in this mode to protect the conflicting outputs and limit the current flow between them.
4. The RDY signal provided by the slave can be read by the master optionally.

Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the COMM[1:0] field in the SPI_CFG2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO or MOSI pin pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode: COMM[1:0] = 01**

  The master in transmit-only mode generates the clock as long as there are data available in the TxFIFO and the master transfer is ongoing.

  The slave in transmit-only mode sends data as long as it receives a clock on the SCK pin and the NSS pin (or software managed internal signal) is active (see *Section 38.4.7*).

- **Receive-only mode: COMM[1:0] = 10**

  In master mode, the MOSI output is disabled and can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled and the CSTART bit in the SPI_CR1 register is set. The clock is stopped either by software explicitly

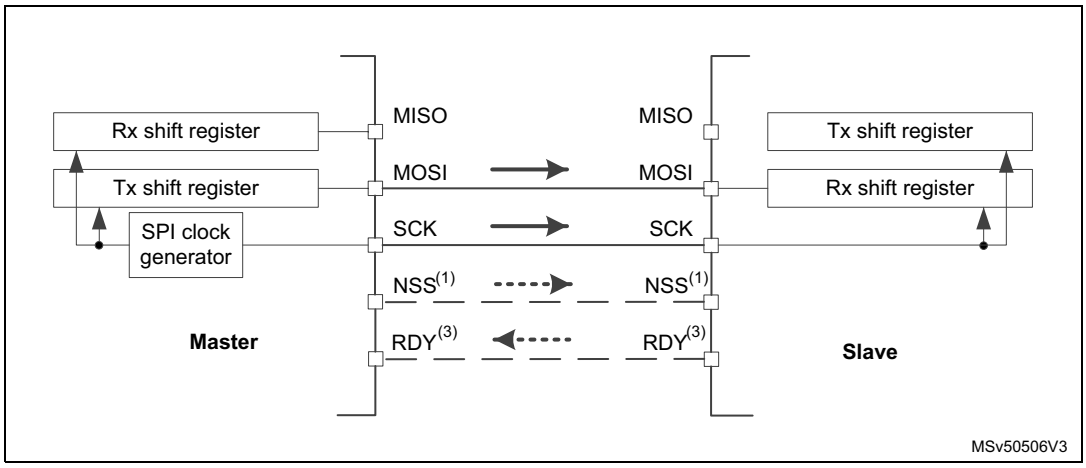
requesting this by setting the CSUSP bit in the SPI_CR1 register or automatically when the RxFIFO is full, when the MASRX bit in the SPI_CR1 is set.

In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see *Section 38.4.7*).

Note: In whatever master and slave modes, the data pin dedicated for transmission can be replaced by the data pin dedicated for reception and vice versa by changing the IOSWP bit value in the SPI_CFG2 register (this bit can only be modified when the SPI is disabled).

Any simplex communication can be replaced by a variant of the half-duplex communication with a constant setting of the transfer direction (bidirectional mode is enabled, while the HDDIR bit is never changed) or by full-duplex control when unused data line and corresponding data flow is ignored.

Figure 346. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode)



1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 38.4.7*).
2. In this configuration, both the MISO pins can be used as GPIOs.
3. The RDY signal provided by the slave can be read by the master optionally.

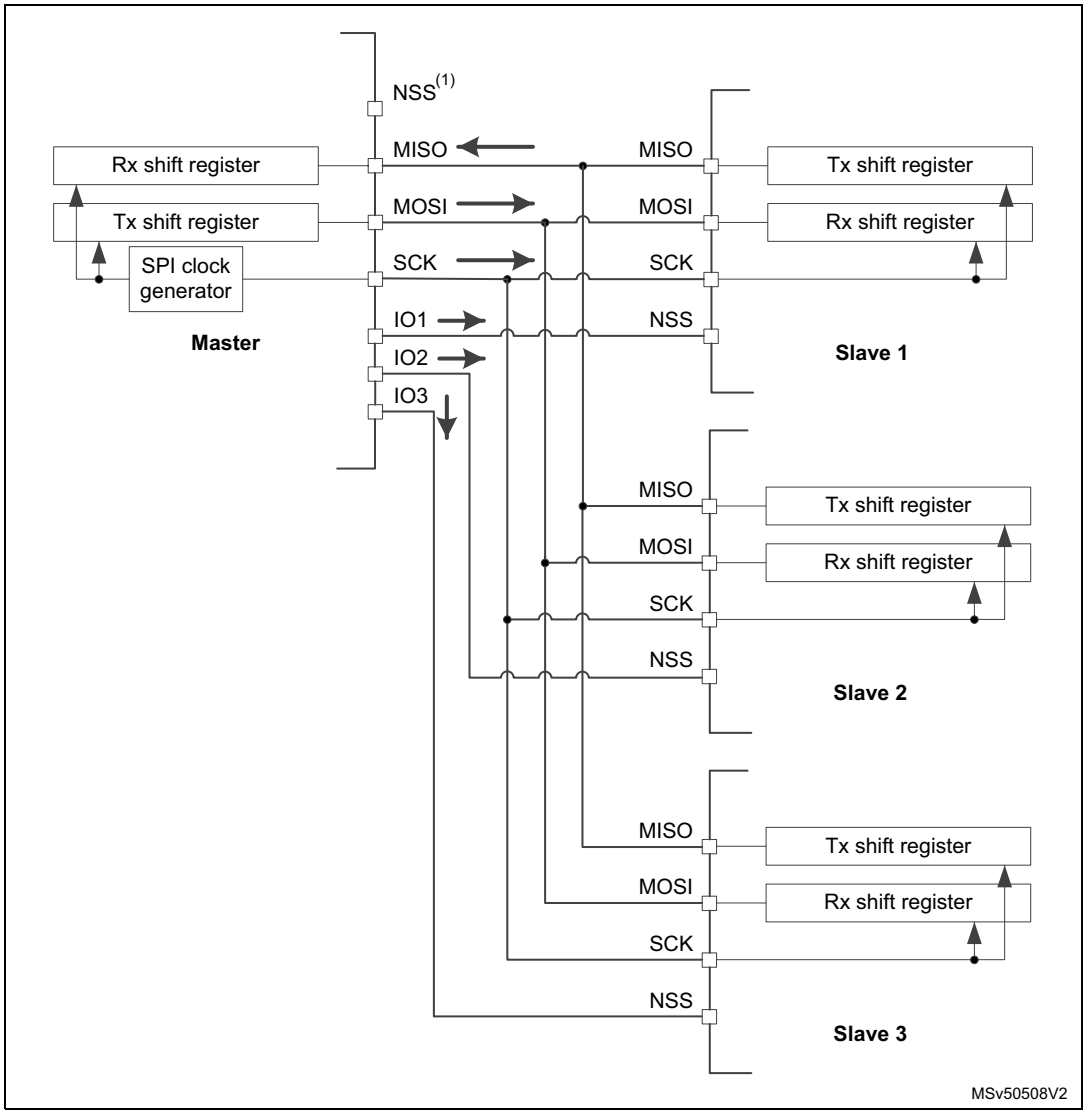
38.4.5 Standard multislave communication

In a configuration with two or more independent slaves, the master uses a star topology with dedicated GPIO pins to manage the chip select lines for each slave separately (see *Figure 347.*).

The master must select one of the slaves individually by pulling low the GPIO connected to the slave NSS input (only one slave can control data on a common MISO line at a given time).

When this is done, a communication between the master and the selected slave is established. In addition to the simplicity, the advantage of this topology is that a specific SPI configuration can be applied for each slave as all the communication sessions are performed separately just within a single master-slave pair. Optionally, when there is no need to read any information from slaves, the master can transmit the same information to the multiple slaves.

Figure 347. Master and three independent slaves connected in star topology



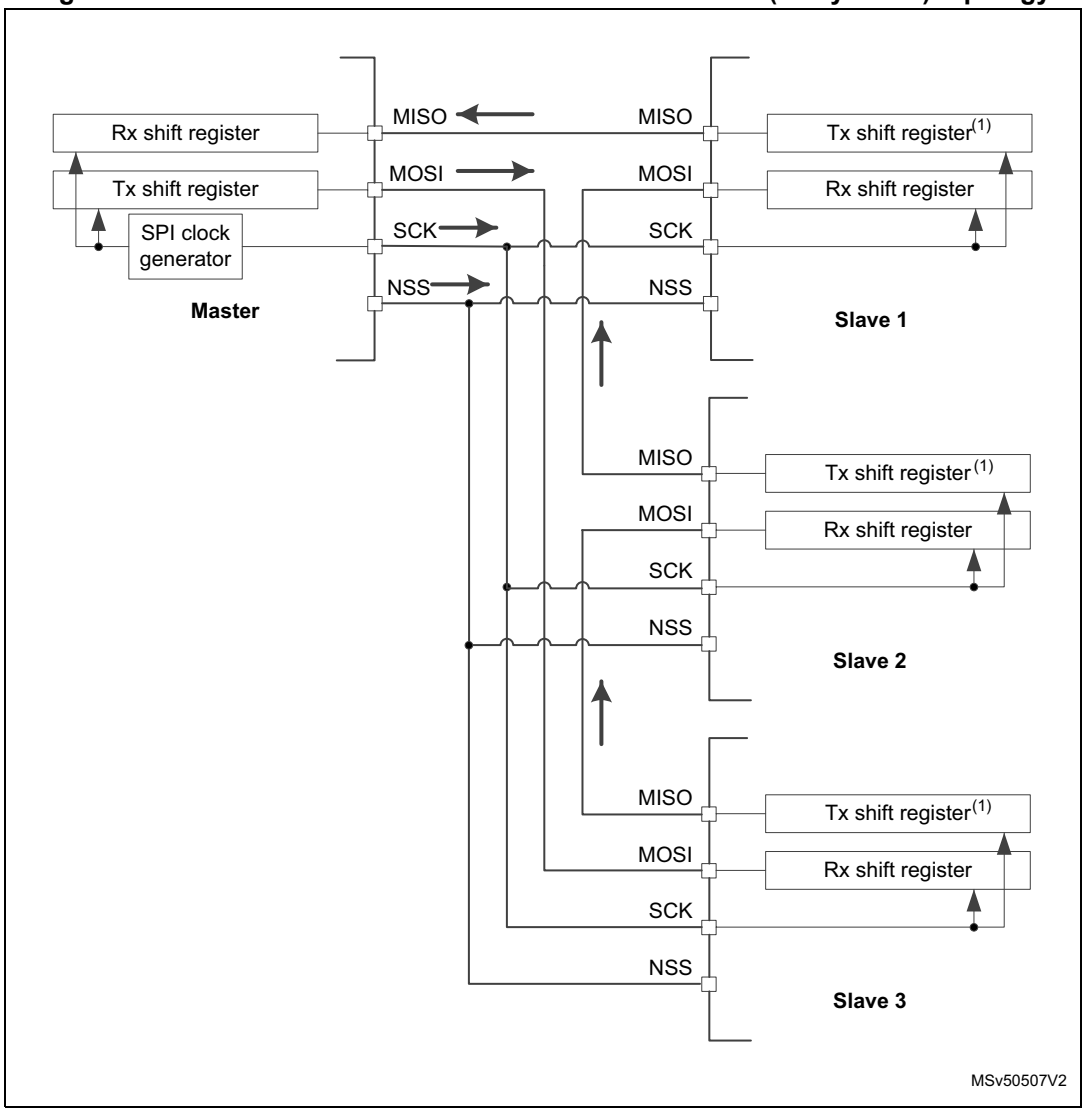
1. The master single NSS pin hardware output functionality cannot support this topology (to be replaced by a set of GPIOs under software control). The NSS pin is free for other application uses (such as GPIO or other alternate functions). Refer to *Section 38.4.7* for details.
2. If the application cannot ensure that no more than a single NSS active signal is provided by the master at a given time, it is better to configure the MISO pins in an open-drain configuration with an external pull-up on the MISO line to prevent conflicts between the interconnected outputs of the slaves. Else, a push-pull configuration can be applied without an extra resistor (see I/O alternate function input/output (GPIO) section).
3. The RDY signals can be read by the master from the slaves optionally.

The master can handle the SPI communication with all the slaves at the same time when a circular topology is applied (see *Figure 348*). All the slaves behave like simple shift registers applied in serial chain under control of common slave select (NSS) and clock (SCK) signals. All the information is shifted simultaneously around the circle while returning back to the master. Sessions have fixed the length where the number of data frames transferred by the master is equal to the number of slaves.

Then when a first data frame is transferred in the chain, the master just sends the information dedicated for the last slave node in the chain, via the first slave node input, while the first information received by the master comes from the last node output at this time.

Correspondingly, the last transferred data finishing the session is dedicated for the first slave node while its first outgoing data just reaches the master input, after its circling around the chain passing through all the other slaves during the session.
The data format configuration and clock setting must be the same for all the nodes in the chain in this topology. As the receive and transmit shift registers are separated internally, a trick with intentional underrun must be applied to the TxFIFO slaves when the information is transferred between the receiver and the transmitter by hardware.
In this case, the transmission underrun feature is configured in a mode repeating the last received data frame (UDRCFG=1). A session can start optionally with a single data pattern written into the TxFIFO by each slave (usually slave status information is applied) before the session starts. In this case, the underrun happens in fact after this first data frame is transferred. To be able to clear the internal underrun condition immediately and restart the session by the TxFIFO content again, the user must disable and enable the SPI between the sessions and must fill the TxFIFO by a new single data pattern (to overcome the propagating delay of the clearing raised in case the underrun is cleared in a standard way by the UDRC bit).

Figure 348. Master and three slaves connected in circular (daisy chain) topology



1. The underrun feature is used by the slaves in this configuration when the slaves are able to transmit data received previously into the Rx shift register once their TxFIFOs become empty.
2. The RDY signals can be read by the master optionally, either separately or configured as open drain outputs (while RDIOP = 0) and connected with a pull-up resistor as a common chain ready status overdriven by the slowest device.

38.4.6 Multimaster communication

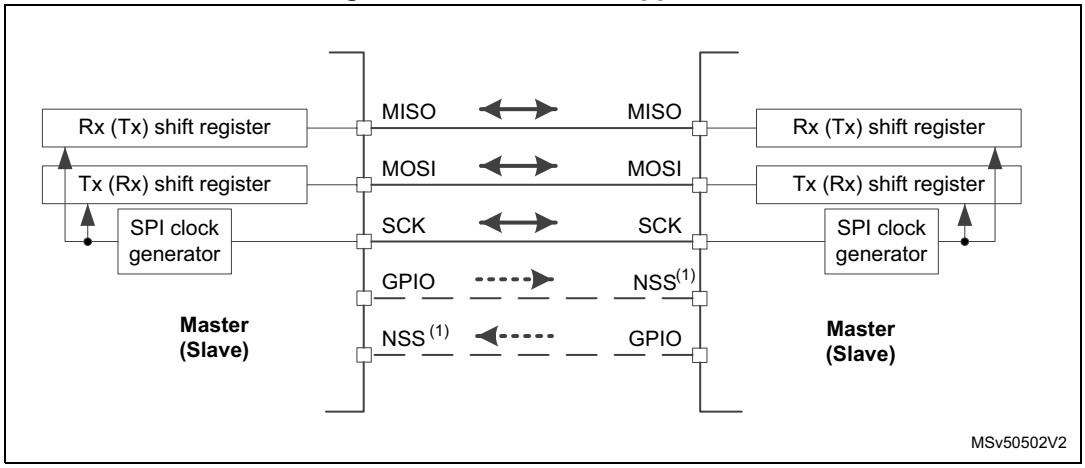
Unless the SPI bus is not designed primarily for a multimaster capability, the user can use a built-in feature that detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, the NSS pin is configured in hardware input mode. The connection of more than two SPI nodes working in this mode is impossible, as only one node can apply its output on a common data line at a given time.

When the nodes are not active, both stay in slave mode by default. Once a node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via the dedicated GPIO pin. After the session is

complete, the active slave select signal is released and the node mastering the bus temporarily returns back to passive slave mode waiting for the next session to start.

If both nodes raise their mastering request at the same time, a bus conflict event appears (see mode fault MODF event). The user can apply some simple arbitration process (for example postpone the next attempt by different predefined timeouts applied to both nodes).

Figure 349. Multimaster application



1. The NSS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control since the passive node is configured as a slave.
2. The RDY signal is not used in this communication.

38.4.7 Slave select (NSS pin) management

In slave mode, the NSS works as a standard ‘chip select’ input and lets the slave communicate with the master. In master mode, the NSS can be used either as an output or an input. As an input it can prevent a multimaster bus collision, and as an output it can drive a slave select signal of a single slave. The NSS signal can be managed internally (software management of the NSS input) or externally when both the NSS input and output are associated with the NSS pin (hardware slave select management). The user can configure which level of this input/output external signal (present on the NSS pin) is considered as active by the SSIOP bit setting. NSS level is considered as active if it is equal to SSIOP.

The hardware or software slave select management can be set using the SSM bit in the SPI_CFG2 register:

- **Software slave select management (SSM = 1)**: in this configuration, slave select information is driven internally by the SSI bit value in the SPI_CR1 register. The external NSS pin is free for other application uses (such as GPIO or other alternate functions).
- **Hardware slave select management (SSM = 0)**: in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in the SPI_CFG2 register).
  - **NSS output enable (SSOE = 1)**: this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The functionality is tied to CSTART and EOT control. As a consequence, the master must apply the proper TSIZE > 0 setting to control the NSS output correctly. Even if SPI AF is not applied at the NSS pin (it can be used as a standard GPIO then), keep anyway SSOE = 1 to ensure the default NSS input level and prevent any mode fault

evaluation at the input of the master NSS internal logic applicable at a multimaster topology exclusively.

a) When SSOM = 0 and SP = 000, the NSS signal is driven to the active level as soon as the master transfer starts (CSTART = 1) and it is kept active until its EOT flag is set or the transmission is suspended.

b) When SP = 001, a pulse is generated as defined by the TI mode.

c) When SSOM = 1, SP = 000 and MIDI > 1 the NSS is pulsed inactive between data frames, and kept inactive for a number of SPI clock periods defined by the MIDI value decremented by one (from 1 to 14).

d) NSS input is forced to nonactive state internally at master to prevent any mode fault.

– **NSS output disable (SSM = 0, SSOE = 0)**:

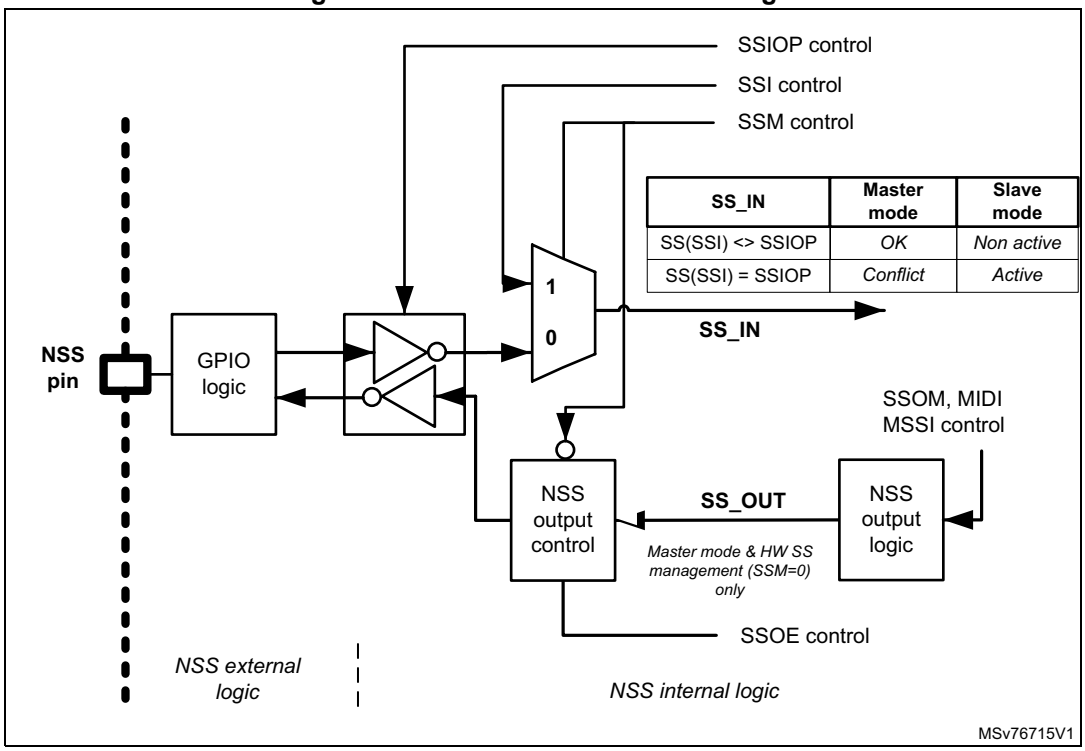
a) If the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled into an active level in this mode, the SPI enters master mode fault state and the SPI device is automatically reconfigured in slave mode (MASTER = 0).

b) In slave mode, the NSS pin works as a standard ‘chip select’ input and the slave is selected while the NSS line is at its active level.

Note: The purpose of automatic switching into slave mode when a mode fault occurs is to avoid the possible conflicts on data and clock lines. As the SPE is automatically reset in this condition, both Rx and Tx FIFOs are flushed and current data is lost.

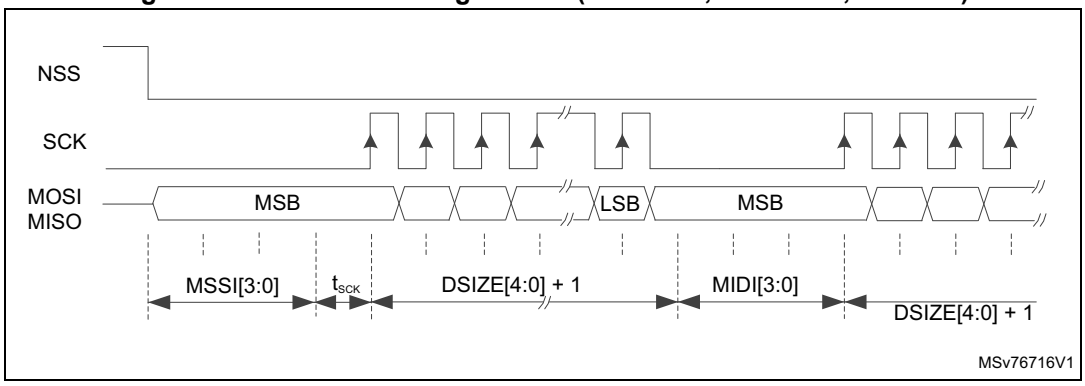
When the SPI slave is enabled in the hardware slave select management mode, all the transfers are ignored even in case of the NSS is found at active level. They are ignored until the slave detects a start of the NSS signal (transition from nonactive to active level) just synchronizing the slave with the master. This is because the hardware management mode cannot be used when the external NSS pin is fixed. There is no such protection in the slave select software management. Then the SSI bit must be changed when there is no traffic on the bus and the SCK signal is in idle state level between transfers exclusively in this case.

Figure 350. Scheme of NSS control logic



When the hardware output NSS control is applied (SSM = 0, SSOE = 1), by configuration of the MIDI[3:0] and MSSI[3:0] bitfields, the user can control the timing of the NSS signal between data frames and can insert an extra delay at the beginning of every transfer (to separate the NSS and clock starts). This can be useful when the slave needs to slow down the flow to obtain sufficient room for correct data handling (see *Figure 351*).

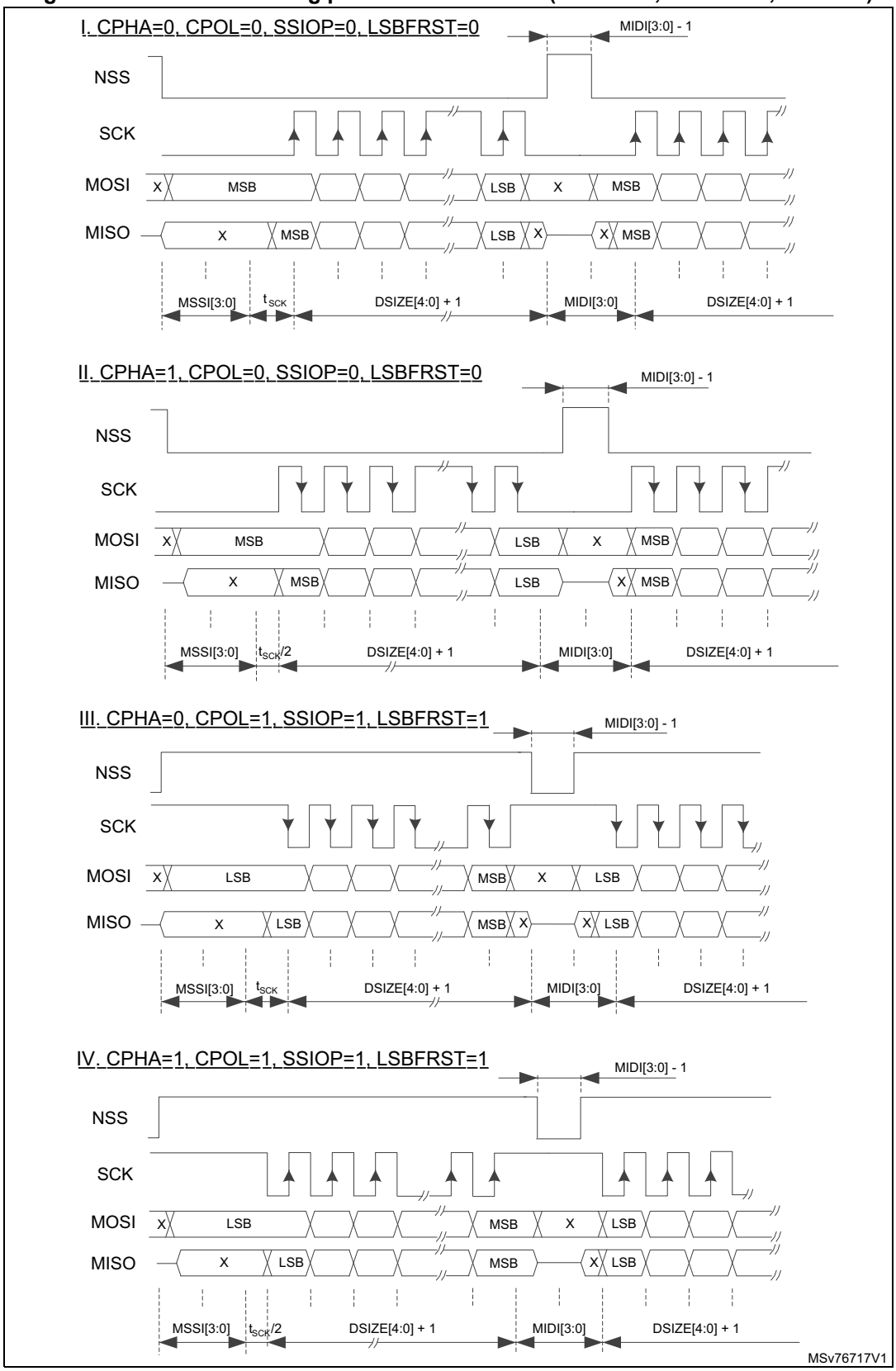
Figure 351. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0)



1. MSSI[3:0] = 0011, MIDI[3:0] = 0011 (SCK flow is continuous when MIDI[3:0] = 0).
2. CPHA = 0, CPOL = 0, SSIOP = 0, LSBFRST = 0.

Additionally, setting the SSOM bit invokes a specific mode, which interleaves pulses between data frames if there is a sufficient space to provide them (MIDI[3:0] must be set greater than one SPI period). Some configuration examples are shown in *Figure 352*.

Figure 352. NSS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0)



1. MSSI[3:0] = 0010, MIDI[3:0] = 0010.
2. NSS interleaves between data when MIDI[3:0] > 1 wide of the interleaving pulse is always one SCK period less than the gap provided between the frames (defined by the MIDI parameter). If MIDI is set, the frames are separated by a single SCK period but no interleaving pulse appears on NSS.

38.4.8 Ready pin (RDY) management

The status of the slave capability to handle data can be checked on the RDY pin. By default, a low level indicates that the slave is not ready for transfer. The reason can be that the slave TxFIFO is empty, RxFIFO full or the SPI is disabled. An active level of the signal can be selected by the RDIOP bit. If the master continues or starts to communicate with the slave when it indicates a not ready status, it is highly probable that the transfer fails.

The logic to control the RDY output is rather complex, tied closely with the TSIZE and DSIZE settings. The RDY reaction is more pessimistic and sensitive to TxFIFO becoming nearly empty and/or RxFIFO nearly full during a frame transfer. This pessimistic logic is suppressed at the end of a transfer only when RDY stays active, despite TxFIFO becomes fully empty and/or RxFIFO becomes fully occupied. The target is to prevent any data corruption and inform the master in time that it is necessary to suspend the transfer temporarily until the next transferred data can be processed safely again. When the RDY signal input is enabled at master side, the master suspends the communication once the slave indicates not ready status. This prevents the master to complete the transfer of an ongoing frame, which just empties the slave TxFIFO or full fills its RxFIFO until a next data is written and/or read there (despite the frame still can be completed without any constraint). It can make a problem if the TSIZE = 0 configuration is applied at slave because slave then never evaluates the end of the transfer (which suppresses the not ready status just when the last data is sent). Then the user has to release the RxFIFO and/or write additional (even dummy) data to TxFIFO by software at slave side to release the not RDY signal, unblock ST master and so enable it to continue at the communication suspended at middle of a frame occasionally.

When RDY is not used by the master, it must be disabled (RDIOM = 0). Then an internal logic of the master simulates the slave status always ready. In this case, the RDIOP bit setting has no meaning.

Due to synchronization between clock domains and evaluation of the RDY logic on both master and slave sides, the RDY pin feature is not reliable and cannot be used when the size of data frames is configured shorter than 8 bits.

38.4.9 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity, and the data frame format. To be able to communicate together, the master and slave devices must follow the same communication format and be synchronized correctly.

Clock phase and polarity controls

Four possible timing relationships can be chosen by software, using the CPOL and CPHA bits in the SPI_CFG2 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transferred (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transferred (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

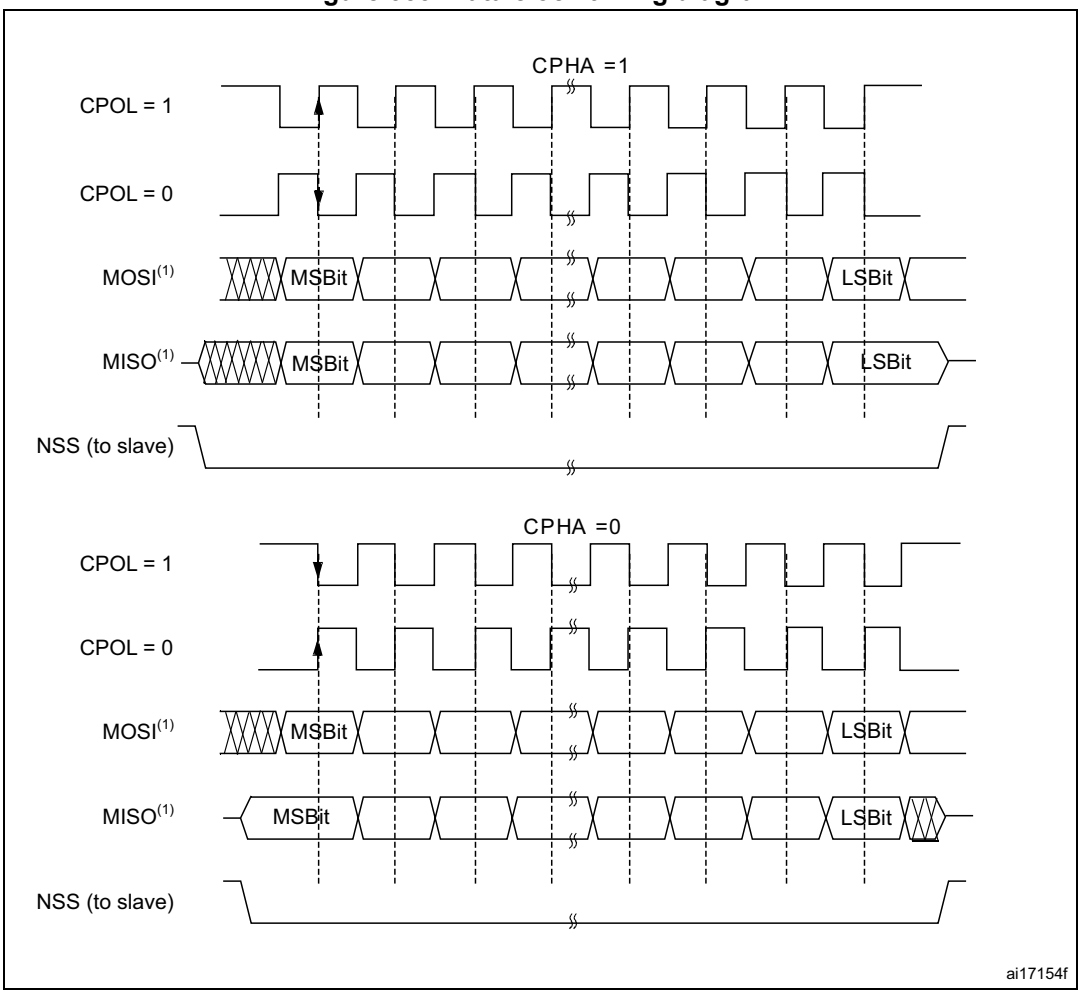
The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edges (dotted lines in *Figure 353*).

Figure 353 shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

Note: Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.

The idle state of SCK must correspond to the polarity selected in the SPI_CFG2 register (by pulling the SCK pin up if CPOL = 1 or pulling it down if CPOL = 0).

Figure 353. Data clock timing diagram



1. The order of data bits depends on the LSBFRST bit setting.

Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFRST bit of the SPI_CFG2 register.

The data frame size is configured through the DSIZE bitfield of the SPI_CFG1 register to a range that depends on the SPI instance (see *Section 38.3: SPI implementation*). The setting applies to both transmission and reception. The data bits in the SPI_TXDR/SPI_RXDR registers are right-aligned with 8, 16, or 32 bits, depending on the data size. These registers can consequently be accessed by bytes, half-words, or words. FIFO accesses smaller than a single data are forbidden. When the FIFO occupancy flag is raised, a FIFO access greater than the FIFO threshold can lead to spurious data being read or written data being lost. When the access to/from the SPI_TXDR/SPI_RXDR registers is a multiple of the configured data size, data packing is applied automatically, while the lowest significant bits/bytes are communicated first. For more details, see *Section 38.4.12: SPI data transmission and reception procedures*.

38.4.10 Configuring the SPI

The configuration procedure is almost the same for the master and the slave. For specific mode setups, follow the dedicated chapters. When a standard communication must be initialized, perform these steps:

1. Write the proper GPIO registers: configure GPIO alternate functions at MOSI, MISO, SCK, NSS, and RDY pins if applied.
2. Write into the SPI_CFG1 and SPI_CFG2 registers and set up the proper values of all ‘not reserved’ bits and bitfields, prior to enabling the SPI, with the following exceptions:
   a) The SSOM, MASRX, SSOE, RDIOM, MBR[2:0], BPASS, MIDI[3:0], MSSI[3:0] bits are taken into account in master mode only, the MSSI[3:0] bits take effect when the SSOE bit is set, the RDIOP bit takes no effect when the RDIOM bit is not set in master mode. When the slave is configured in TI mode, the MBR[2:0] setting is also considered.
   b) UDRCFG is taken into account in slave mode only.
   c) CRCSIZE[4:0] bitfield must be configured if CRCEN is set.
   d) CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI, and MIDI are not required in TI mode.
   e) Once the AFCNTR bit is set in the SPI_CFG2 register, all the SPI outputs start to be propagated onto the associated GPIO pins regardless of the peripheral enable. So, any later configuration changes of the SPI_CFG1 and SPI_CFG2 registers can affect the level of signals on these pins.
3. Write to the SPI_CR2 register to select the length of the transfer, if it is not known TSIZE must be programmed to zero.
4. Write to the SPI_CRCPOLY and into the TCRCINI, RCRCINI, and CRC33_17 bits of the SPI_CR1 register to configure the CRC polynomial and CRC calculation if needed.
5. Configure DMA streams dedicated for the SPI Tx and Rx in DMA registers if the DMA streams are used (see *Section 38.4.14: Communication using DMA (direct memory addressing)*).
6. Configure SSI, HDDIR, and MASRX in the SPI_CR1 register if required.
7. Program the IOLOCK bit in the SPI_CR1 register if the configuration protection is required (for safety).

38.4.11 Enabling the SPI

It is recommended to configure and enable the SPI slave before the master sends the clock. But there is no impact if the configuration and enabling procedure is done while traffic is ongoing on the bus, assuming that the NSS signal is managed by hardware at slave or kept inactive by the slave software when the software management of the NSS signal is applied (see *Section 38.4.7*). To prevent any risk of any data underrun, all the data to be sent have to be written to the slave transmitter data register before the master starts its clocking. The SCK signal must be settled to the idle state level corresponding to the selected polarity, before the SPI slave is selected by NSS, else the following transfer may be desynchronized.

When the SPI slave is enabled at the hardware slave select management mode, all the transfers are ignored even in case of the NSS is found at active level. They are ignored until the slave detects a start of the NSS signal (its transition from nonactive to active level) just synchronizing the slave with the master. That is why the hardware management mode cannot be used when the external NSS pin is fixed. There is no such protection at the slave select software management. In this case, the SSI bit must be changed when there is no traffic on the bus and the SCK signal is at idle state level between transfers exclusively in this case.

The master in full duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled, the CSTART bit is set, and the TxFIFO is not empty, or with the next write to TxFIFO.

In any master receive-only mode, the master starts to communicate and the clock starts running after the SPI is enabled and the CSTART bit is set.

For handling DMA, see *Section 38.4.14*.

38.4.12 SPI data transmission and reception procedures

The SPI can transfer data at a very high communication speed. Even though the data is FIFO buffered, handling a continuous data flow by servicing data frames individually leads to an enormous CPU or DMA load, in particular when the data size is small. This potentially leads to a significant limitation of the overall system performance, as well as communication errors such as data overrun or underrun that may be raised when the system latencies in servicing requests become comparable to single data frame transfer time.

The SPI offers advanced hardware control features to prevent these issues from occurring:

- Decreasing the number of events requiring service by collecting the data frames into larger **data packets**

  The number of data frames per packet (FIFO threshold) can be configured. The complete data packet generates a FIFO occupancy event, which is then handled within a single, compact, service.
- Decreasing the number of CPU execution cycles required for the service, by applying the appropriate **data access**:
  - The widest access to the SPI data registers must be applied. This allows cumulative data to be handled by a single register read or write operation (data packing).
  - The number of read/write accesses necessary to complete the data packet must be aligned with the data frame size and the FIFO threshold.
- **Concurrent read and write services** to handle full-duplex data flow based on common FIFO occupancy events.
- Embedded **hardware data counters** to define the exact number of data involved in transfers.

Note: Using these features is optional. Nothing prevents the application from handling the data flow frame by frame.

The following sections give more details and describe specific cases for using these advanced hardware control features to handle data transfers.

Data packet control

The data frame size (number of bits in the frame) is defined through the DSIZE bitfield of the SPI_CFG1 register. The number of data frames per packet can be configured by selecting the FIFO threshold through the FTHLV bitfield of the SPI_CFG1 register. If the threshold value is set to zero, each completed data frame raises the FIFO occupancy event. The data packet occupancy must not exceed half of the FIFO size, which depends on the SPI instance of the product. The FIFO capacity (multiple of 8 bits) is consumed according to the following rules, depending on the data frame size:

- Data frames from 4 to 8 bits occupy one byte of the FIFO.
- Data frames from 9 to 16 bits occupy two bytes of the FIFO.

For example, if the FIFO capacity is 16 bytes, it accommodates up to five 24-bit frames. In this case, the data packet configuration must not exceed two data frames.

The SPI features two separate FIFOs to handle the reception and transmission data flow, the RxFIFO, and the TxFIFO. Their content can be handled by monitoring the occupancy

flags RXP, TXP, and DXP of the SPI_SR register according to the SPI mode (duplex or simplex):

- If RXP is set, at least one complete data packet can be read from the RxFIFO.
- If TXP is set, at least one complete data packet can be written to the TxFIFO.
- If DXP is set, at least one complete data packet can be read from the RxFIFO and written to the TxFIFO at full-duplex mode.

These flags can be polled by software, or they can trigger an interrupt and/or a DMA request (if enabled through the RXPIE, TXPIE, and DXPIE bits of the SPI_EIR register or the RXDMAEN and TXDMAEN bits of the SPI_CFG1 register).

Once an occupancy flag is set, the software or the DMA must read and/or write a complete data packet before checking the flag again to verify if the next packet can be handled. This cycle can be repeated until the corresponding flag is read at zero.
Both RxFIFO and TxFIFO contents are flushed and cannot be accessed when the SPI is disabled (SPE cleared in the SPI_CR1 register).

Data packing versus data register access control

The content of the RxFIFO and TxFIFO can be accessed by reading/writing from/to the SPI data registers SPI_RXDR and SPI_TXDR, respectively. A read access from the SPI_RXDR register returns the oldest values stored in the RxFIFO that have not been read yet. A write access to the SPI_TXDR register stores the data written at the end of the send queue in the TxFIFO.

These data registers can be accessed by 8-, 16-, or 32-bit read and write CPU instructions, forced by the register address casting applied by the software. Data packing is performed automatically by hardware if the data access applied by the software is a multiple of the data size. It allows handling more than one data in parallel in a single data register access. Then the SPI operates using the lowest significant byte or half-word first. FIFO data accesses of less than the configured data size are forbidden. To avoid spurious data being read or written data being lost, a complete data packet (configured through the FIFO threshold) must be serviced when the corresponding FIFO occupancy flag is set.

For example, if the data frame size fits into one byte, the data packing is used automatically when a 16-bit or 32-bit read/write access is performed by software from/to the SPI_RXDR/SPI_TXDR register. In this case, two respectively four data are handled by a single 16-bit respectively 32-bit access. Additionally, if such data frames are grouped into packets of four via the FIFO threshold setting, the packet to be serviced is signalized by raising the corresponding FIFO occupancy flag (TXP, RXP, or DXP). If the instance FIFO threshold is sufficient, a packet containing eight 8-bit data frames can be handled by two consecutive 32-bit accesses upon the same single threshold event. The most efficient data handling is achieved when the data packet size (defined by DSIZE and FTHLV bitfields) is aligned with the read/write access from/to the data registers.

Concurrent read and write services

In full-duplex mode, both TxFIFO and RxFIFO packet occupancies can be monitored through a common FIFO flag (DXP). When the DXP flag is set, the application performs the specified number of writes to SPI_TXDR to upload the content of one entire data packet for transmission, followed by the same number of reads from SPI_RXDR to download the content of one received data packet. Once one data packet is uploaded and one packet is downloaded, the application software or the DMA checks again the DXP flag and repeats the data packet read and write operation sequence until DXP is read as zero.

The drawback of services based on DXP exclusively is that servicing the TxFIFO is delayed on purpose due to the SPI nature since the TXP events precedes the RXP ones. To allow continuous SCK clock flow on the master side and prevent underrun on the slave side, it is recommended to prefill a few data ahead to the TxFIFO at the transfer start, and wait until the transfer is complete to read the last received data.

Hardware data counter

If a hardware data counter is used (TSIZE bitfield of the SPI_CR2 register set to a nonzero value), the application software does not need to calculate the remaining number of data to be handled. The end of transfer automatically controls the CRC, as well as the hardware slave select management, when used. The user application does not need to ensure that the overall number of data is a multiple of the packet size. If the last data packet is incomplete, it is serviced in the same way as any full packet. The unused part of this last packet is not handled by the peripheral. Only the valid data are written into the TxFIFO and/or read from the RxFIFO, and the redundant writes and reads are discarded. When the hardware counter reaches zero, the EOT flag is raised, and the RXP flag is not set for the last data packet. If the last packet is not aligned with the packet size, the TXP and EOT occupancy events are not related to the configured packet size but to the number of remaining data calculated by hardware.

If TSIZE is kept at zero (for example due to an unpredictable number of data), only the number of transferred data corresponding to the FIFO thresholds is supported. If some data are not aligned with the configured packet size, they may remain pending and available in the RxFIFO. In this case, the FTHLV level is not reached and the RXP flag is not set. Then the number of remaining received data frames in the RxFIFO is indicated by the RXWNE and RXPLVL bitfields of the SPI_SR register. Nevertheless, the application software can still read the complete data packet from the RxFIFO and the redundant data are read as zero. To prevent such an unaligned data reception, the user must configure the FIFO threshold to a single data (FTHLV = 0). In this case, each data frame is serviced by its own RXP occupancy event.

Data transfer handling

Data are transferred using MOSI and MISO lines, depending on the SPI communication mode and associated configuration. The mode is configured through the COMM[1:0] bitfield of the SPI_CFG2 register. The HDDIR bit of the SPI_CR1 register controls the data flow direction in half-duplex mode. The communication flow is handled by the master via the NSS and SCK signals. The slave selected for the communication is fully subordinated to the master communication activity, no matter if it handles the data flow on time or not. The slave can only temporarily suspend the master communication by using the RDY signal. The active levels of RDY and NSS signals can be changed, or MISO and MOSI functionality swapped (via the RDIOP, SSIOP and IOSWP bits of the SPI_CFG2 register).

The SPI master transmitter can operate in full-duplex, simplex, or half-duplex mode. In full-duplex mode, data are received synchronously. The master starts the data transfer once the CSTART bit of the SPI_CR1 register is set, provided the SPI is enabled and the TxFIFO content is not empty. The master then provides the serial clock signal continuously on the SCK pin until:

- The total number of required data programmed in TSIZE is transferred, or
- The transfer is suspended.

An automatic temporary suspension of the master transmission occurs when:

- The slave does not assert the RDY signal, if it is used, or
- The TxFIFO becomes empty, or
- In full-duplex mode, the RxFIFO becomes full while the automatic suspension is enabled (MASRX set in the SPI_CR1 register).

When an automatic suspension occurs, the master stops providing the clock, and the transfer proceeds depending on the cause of the suspension, when:

- The slave asserts RDY.
- The master software or the DMA writes additional data to TxFIFO.
- In full-duplex mode, the master software or the DMA releases the RxFIFO to enable it to accommodate new data.

The SPI master can receive data only when it operates in simplex receiver or half-duplex receiver mode. In these modes, the master starts the data transfer when the SPI is enabled, and the transfer is released by setting the CSTART bit. The serial clock signal is then provided continuously on the SCK pin by the master until:

- The total number of required data programmed in TSIZE is received, or
- The transfer is suspended by the master.

An automatic temporary suspension of the reception occurs when:

- The slave does not assert the RDY signal, if it is used, or
- The RxFIFO becomes full while the automatic suspension is enabled (MASRX bit set in the SPI_CR1 register).

When an automatic suspension occurs, the master stops providing the clock and the transfer proceeds depending on the cause of the suspension, when:

- The slave asserts RDY.
- The master software or the DMA releases the RxFIFO to enable it in order to accommodate new data.

A preferable way to terminate a transfer is to program the TSIZE bitfield to generate an EOT event. Hardware NSS signal or CRC handshake can then be controlled. To restart the internal state machine properly, it is recommended to disable the SPI and enable it again when the transfer is complete, even if the SPI configuration is not changed.

A transfer can be suspended at any time by setting the CSUSP bit of the SPI_CR1 register, which clears the CSTART bit. This software suspension control ensures the completion of any ongoing data frame. To restart the internal state machine properly, the SPI must be disabled and enabled again before the next transfer starts.

Oppositely, a temporary automatic suspension controlled by hardware typically results in a frozen and incomplete data frame transfer. This depends on baud rate setting, but usually, due to internal synchronization delays, the SCK signal stops when a few bits from the next data frame are already transferred on the bus. Once the suspension is released, the data frame transmission completes by transferring the remaining bits. That is why this automatic suspension is not quite reliable when the data frame size is less than eight bits. When shorter data frames are used, to prevent any data loss and assure proper RDY and/or MASRX operation, the user can interleave and so extend the transfer time by inserting additional dummy clock cycles so that the period for a single data frame is higher or equal to eight SCK duration. This is done by setting the MIDI[3:0] bitfield of the SPI_CFG2 register.

Caution: If the SPE bit is cleared in master mode while the transfer is ongoing without any suspension, the clock is stopped even if the current frame transmission is not complete, and the content of the FIFOs is flushed and lost.

38.4.13 Disabling the SPI

To disable the SPI, it is mandatory to follow the disable procedures described in this paragraph.

In the master mode, it is important to do this before the system enters a low-power mode when the peripheral clock is stopped, otherwise, ongoing transfers may be corrupted.

In slave mode, the SPI communication can continue when the **spi_pclk** and **spi_ker_ck** clocks are stopped, without interruption, until any end of communication or data service request condition is reached. The **spi_pclk** can generally be stopped by setting the system into Stop mode. Refer to the RCC section for further information.

The master in full-duplex or transmit-only mode can finish any transfers when it stops providing data for transmission. In this case, the clock stops after the last data transfer. TXC flag can be polled (or interrupt enabled with EOTIE = 1) in order to wait for the last data frame to be sent.

When the master is in any receive-only mode, to stop the peripheral, the SPI communication must first be suspended, by setting the CSUSP bit.

The data received but not read remain stored in RxFIFO when the SPI is suspended.

After such a software suspension, SPI must always be disabled to restart the internal state machine properly.

When SPI is disabled, RxFIFO is flushed. To prevent losing unread data, the user must ensure that RxFIFO is empty when disabling the SPI, by reading all remaining data (as indicated by the RXP, RXWNE, and RXPLVL fields in the SPI_SR register).

The standard disable procedure is based on polling EOT and/or TXC status to check if a transmission session is (fully) completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transfers, for example:

- When the master handles the NSS signal by a GPIO not related to SPI (for example at case of multislave star topology) and it has to provide a proper end-of-NSS pulse for the slave, or
- When transfers from DMA or FIFO are completed while the last data frame or CRC frame transfer is still ongoing in the peripheral bus.

When TSIZE > 0, EOT and TXC signals are equal so polling of EOT is reliable at whatever SPI communication mode to check the end of the bus activity. When TSIZE = 0, the user has to check TXC, SUSP, or FIFO occupancy flags according to the applied SPI mode and the way of the data flow termination.

The correct disable procedure in master mode, except when receive-only mode is used, is:

1. Wait until TXC = 1 and/or EOT = 1 (no more data to transmit and last data frame sent). When CRC is used, it is sent automatically after the last data in the block is processed. TXC/EOT is set when the CRC frame is complete in this case. When a transmission is suspended the software has to wait until the CSTART bit is cleared.
2. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
3. Disable the SPI (SPE = 0).

The correct disable procedure for master receive-only modes is:

1. Wait on EOT or break the receive flow by suspending SPI (CSUSP = 1).
2. Wait until SUSP = 1 (the last data frame is processed) if the receive flow is suspended.
3. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
4. Disable the SPI (SPE = 0).

In slave mode, any ongoing data are lost when disabling the SPI.

Controlling the I/Os

As soon as the SPI is disabled, the associated and enabled AF outputs can still be driven by the device depending on the AFCNTR bit of the SPI_CFG2 register. When active output control is applied (AFCNTR = 1) and SPI has just been disabled (SPE = 0), the enabled outputs associated with SPI control signals (like NSS and SCK at master and RDY at slave) can immediately toggle to inactive level (according to SSIOP and CPOL settings at master and RDIOP at slave respectively). Instead, the data line output (MOSI at master and MISO at slave) can immediately change its level depending on the actual TxFIFO content, with the effect of potentially making invalid and no more guaranteed the value of the latest transferred bit on the bus. If necessary, the user has to take care about proper data hold time at the data line and avoid any eventual fast SPI disable just after the last data transfer is complete.

Note: Despite stability of the latest bit is guaranteed by design during the sampling edge of the clock, some devices can require even extension of this data bit stability interval during the sampling. It can be done, for example by inserting a small software delay between EOT event occurrence and SPI disable action.

38.4.14 Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXDMAEN or RXDMAEN enable bits of the SPI_CFG1 register are set. Separate requests must be issued to the Tx and Rx buffers to fulfill the service of the defined packet.

- In transmission, a series of DMA requests is triggered each time TXP is set. The DMA then performs a series of writes to the SPI_TXDR register.
- In reception, a series of DMA requests is triggered each time RXP is set. The DMA then performs a series of reads from the SPI_RXDR register. When EOT is set at the end of the transfer and the last data packet is incomplete, then DMA request is activated automatically according to RXWNE and RXPLVL[1:0] setting to read the rest of data.

If the SPI is programmed in receive-only mode, UDR is never set.

If the SPI is programmed in a transmit mode, TXP and UDR can be eventually set at slave side, because transmit data may not be available. In this case, some data are sent on the TX line according with the UDR management selection.

When the SPI is used at a simplex mode, the user must enable the adequate DMA channel only while keeping the complementary unused channel disabled.

If the SPI is programmed in transmit-only mode, RXP and OVR are never set.

If the SPI is programmed in full-duplex mode, RXP and OVR are eventually set, because received data are not read.

In transmission mode, when the DMA or the user has written all the data to be transmitted (the TXTF flag is set in the SPI_SR register), the EOT (or TXC at case TSIZE = 0) flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or before disabling the **spi_pclk** in master mode. The software must first wait until EOT = 1 and/or TXC = 1.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:

1. Enable the DMA Rx buffer in the RXDMAEN bit of the SPI_CFG1 register, if DMA Rx is used.
2. Enable DMA requests for Tx and Rx in DMA registers, if the DMA is used.
3. Enable the DMA Tx buffer in the TXDMAEN bit of the SPI_CFG1 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication, it is mandatory to follow these steps in order:

1. Disable DMA requests for Tx and Rx in the DMA registers, if the DMA issued.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits of the SPI_CFG1 register, if DMA Tx and/or DMA Rx are used.

Data packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPI_CFG1 register) the packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel.

The packing mode is enabled if the DMA channel PSIZE value is a multiple of the data size. Then the DMA automatically manages the sequences of write and read operations to/from the SPI data registers, based on FIFO occupancy flags, and depending on the FIFO threshold and data size configurations.

The DMA completes the transfer automatically according to the TSIZE field setting, whatever the data packing mode used, and even if the number of data to transfer is not a multiple of the DMA data size (16 bits or 32 bits) while the frame size is smaller.

Alternatively, the last data frames can be written by software, in the single/unpacked mode.

Configuring any DMA data access to less than the configured data size is forbidden. One complete data frame must be always accessed at minimum.

38.4.15 Autonomous mode

The SPI is capable of handling and initializing transfers autonomously. It requires no specific system execution interaction until the ongoing transfer ends.

Such autonomous transfers can be handled not only in Run or Sleep mode but even in Stop mode when the SPI logic is able to provide temporary clock requests addressed to the reset and clock controller (RCC) to ensure clocking of those SPI domains just necessary for handling the data flow between the memory and the peripheral interface at dependency in the SPI mode.

In Stop mode, the APB clock is requested by the peripheral each time the SPI registers need to be updated based on specific traffic events (mainly TXP and RXP). The required

clock is provided by RCC if SPI autonomous mode is enabled at the RCC configuration and the SPI is clocked by an internal oscillator available in Stop mode.

Interrupts or DMA requests are then generated, depending on the SPI configuration. If no interrupt is enabled, the device remains in Stop mode. If DMA requests are enabled, the data are directly transferred to/from the SRAM thanks to the DMA while the device remains in Stop mode. If an enabled interrupt occurs, the device wakes up from Stop mode.

Note: The peripheral clock request stays pending until the flag with enabled interrupt is set. This is why it is important to service these pending requests and clear their flag as soon as possible at system sensitive to the low-power consumption especially, and the application must acknowledge all pending interrupts events before switching the SPI to low-power mode.

Slave mode

When the SPI is configured as a standard slave and the device is at Stop mode, the SPI kernel clock and the SPI APB clock are not provided permanently. All the data flow between the SPI interface and associated FIFOs is handled by an external SCK clock provided by the outer master device within the serial interface clock domain. APB clock temporal requests are then based upon specific traffic events at dependency on the SPI configuration. As the slave never initializes a transfer, there is no need to synchronize any transfer start in this mode.

Note: The peripheral clock request stays pending until the flag with enabled interrupt stays set. This is why it is important to service these pending requests and clear their flag as soon as possible to achieve the lowest power consumption, and the application must acknowledge all pending interrupt events before switching the SPI to low-power mode.

Master mode

The SPI operating in master mode provides the SCK signal for outer slaves until the transfer is complete. The SCK signal is derived from the SPI clock generator running within the kernel clock domain fed from the RCC upon kernel clock request provided by the SPI when the device is in Stop mode. Temporal requests for the APB clock are then based upon specific traffic events, depending on the SPI configuration. The SPI master always initializes a transfer.

To minimize consumption in Stop mode, it is suggested to combine communication starts triggered by hardware (the TRIGEN bit is set in the SPI_AUTOCR register) and transfers of predefined data size (TSIZE > 0 in the SPI_CR2 register). This ensures that any APB clock request is suppressed between EOT handling and the next trigger event.

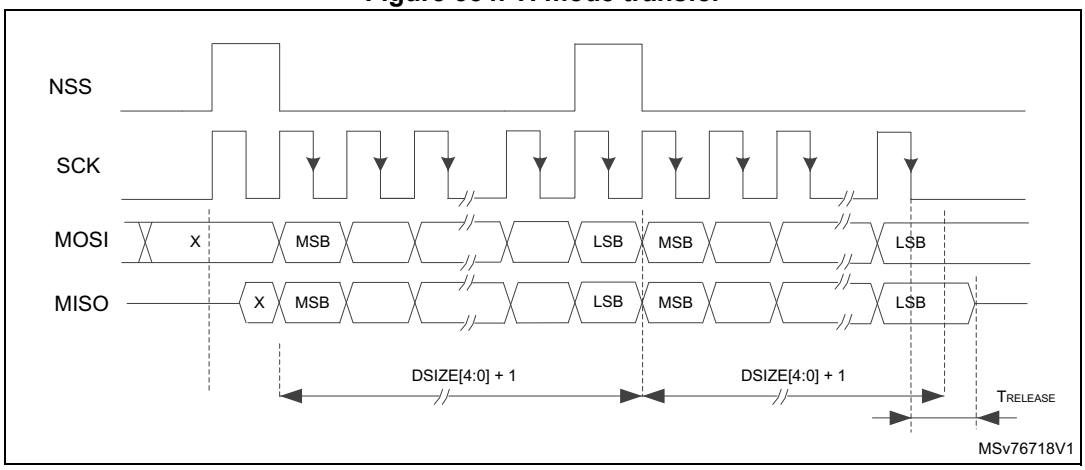
A transfer starts once the CSTART bit is set. In the case of master transmitter, the TxFIFO must be filled by data, too. The CSTART bit can be written either by software or by hardware when a synchronous trigger is detected at Run, Sleep, or Stop mode. The trigger source is selected by the TRIGSEL bits and enabled by the TRIGEN bit of the SPI_AUTOCR register. When the enabled trigger is detected, the transfer starts and continues by handling data until the EOT event or the transfer suspension. When the TRIGEN bit is changed, the user must prevent any trigger event occurrence. If the user cannot prevent that, the TRIGEN bit must be written while the SPI is disabled otherwise the peripheral behavior is not guaranteed.

38.5 SPI specific modes and control

38.5.1 TI mode

With a specific SP[2:0] bitfield setting of the SPI_CFG2 register, the SPI can be configured compliant with the TI protocol. The SCK and NSS signals polarity, phase and flow, as well as the bit order are fixed, so the setting of CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI, and MIDI is not required when the SPI is in TI mode configuration. The NSS signal synchronizes the protocol by pulses over the LSB data bit as it is shown in *Figure 354*.

Figure 354. TI mode transfer



In slave mode, the clock generator is used to define the time when the slave output at MISO pin becomes high-Z when the current transfer finishes. The master baud rate setting (MBR[2:0] of SPI_CFG1) is applied and any baud rate can be used to determine this moment with optimal flexibility. The delay for the MISO signal to become high-Z (TRELEASE) depends on internal resynchronization, too, which takes the next additional 2-4 periods of the clock signal feeding the generator. It is given by the following formula:

$$\frac{Tbaud}{2} + 2 \times Tspi_ker_ck \leq Trelease \leq \frac{Tbaud}{2} + 4 \times Tspi_ker_ck$$

If the slave detects a misplaced NSS pulse during a data transfer the TIFRE flag is set.

38.5.2 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and the interrupt is enabled by setting the corresponding interrupt enable bit.

Overrun flag (OVR)

An overrun condition occurs when data are received by a master or slave and the RxFIFO has not enough space to store these received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RxFIFO).

When an overrun condition occurs, the OVR flag is set and the newly received value does not overwrite the previous one in the RxFIFO. The newly received value is discarded and all

data transmitted subsequently are lost. The OVR flag triggers an interrupt if the OVRIE bit is set. Clearing the OVR bit is done by setting the OVRC bit of the SPI_IFCR register. Clearing the RxFIFO content by performing software reads before the OVR bit is cleared reduces the risk of any immediate repetition of its next overrun. It is suggested to release the RxFIFO space as much as possible, this means to read out all the available data packets based on RXP flag indication.

In master mode, the user can prevent the RxFIFO overrun by automatic communication suspend (MASRX bit).

Underrun flag (UDR)

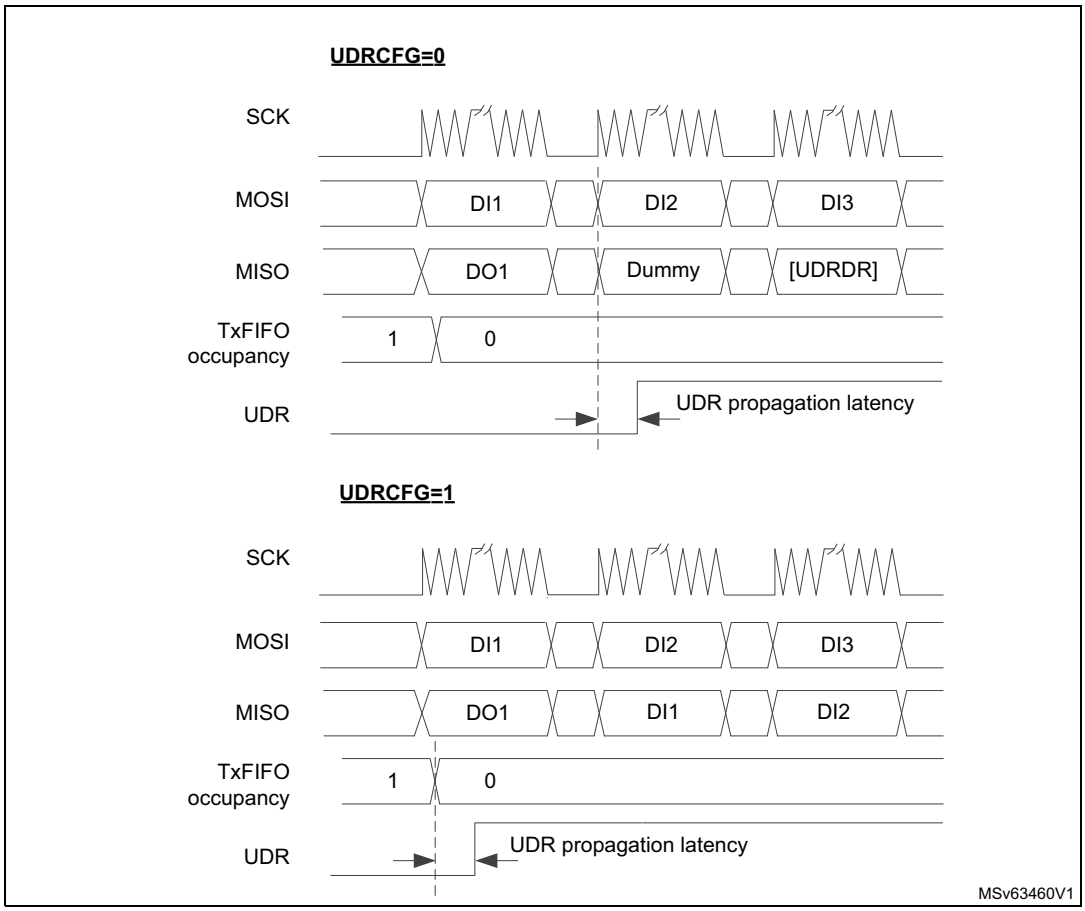
In a slave-transmitting mode, the underrun condition is captured internally by hardware if no data is available for transmission in the slave TxFIFO commonly. The UDR flag setting is then propagated into the status register by hardware (see note below). UDR triggers an interrupt if the UDRIE bit is set.

Underrun detection logic and system behavior depend on the UDRCFG bit. When an underrun is detected by the slave, it can provide out either a constant pattern stored by the user at the UDRDR register or the data received previously from the master. When the first configuration (UDRCFG = 0) is applied, the underrun condition is evaluated whenever the master starts to communicate a new data frame while TxFIFO is empty. Then single additional dummy (accidental) data is always inserted between the last valid data and the constant pattern defined at the UDRDR register (see *Figure 355*). The second configuration (UDRCFG=1) can be used in circular topography structures (see *Figure 348*). Assuming that TxFIFO is not empty when the master starts the communication, the underrun condition is evaluated just once the FIFO becomes empty during the next data flow. Valid data from TxFIFO is then upended by the last received data immediately.

The standard transmission is reenabled once the software clears the UDR flag and this clearing is propagated into SPI logic by hardware. Writing some data to the TxFIFO before the UVR bit is cleared reduces the risk of any immediate repetition of its next underrun.

The data transferred by the slave is unpredictable especially when the transfer starts or continues while TxFIFO is empty and the underrun condition is either not yet captured or just cleared. Typically, this is the case when SPI is just enabled or when a transfer with a defined size just starts. First bits can be corrupted in this case, as well, when the slave software writes the first data into the empty TxFIFO too close prior to starting the data transfer (propagation of the data into TxFIFO takes a few APB clock cycles).

Figure 355. Optional configurations of the slave behavior when an underrun condition is detected



Note: The hardware propagation of an UDR event needs additional traffic on the bus. It always takes a few extra SPI clock cycles after the event happens (both underrun captured by hardware and cleared by software). If clearing of the UDR flag by software is applied close to the end of the data frame transfer or when the SCK line is at idle in between the frames, the next extra underrun pattern is sent initially by the slave before the valid data from TxFIFO becomes transferred again. The user can prevent this by SPI disable/enable action between sessions to restart the underrun logic and so initiate the next session by the valid data.

Mode fault (MODF)

Mode fault occurs when the master device has its internal NSS signal (NSS pin in slave select hardware mode, or SSI bit in slave select software mode) pulled low. This automatically affects the SPI interface in the following ways:

- The MODF bit is set and the SPI interrupt is triggered if the MODFIE bit is set.
- The SPE bit is forced to zero until the MODF bit is set. This disables the SPI and blocks all the peripheral outputs except the MODF interrupt request if enabled.
- The MASTER bit is cleared, thus forcing the device into slave mode.

MODF is cleared by writing 1 to the MODFC bit of the SPI_IFCR register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled to its nonactive level before reenabling the SPI, by setting the SPE bit.

As a security, the hardware does not allow the SPE bit to be set while the MODF bit is set. In a slave device, the MODF bit cannot be set except as the result of a previous multimaster conflict.

A correct software procedure when a master overtakes the bus in multimaster systems must be the following one:

1. Switch into master mode while SSOE = 0 (potential conflict can appear when another master occupies the bus. In this case, MODF is raised, which prevents any next node switching into master mode).
2. Put GPIO pin dedicated for another master slave select control into active level.
3. Perform a data transfer.
4. Put GPIO pin dedicated for another master slave select control into nonactive level.
5. Switch back to slave mode.

CRC error (CRCE)

This flag is used to verify the validity of the value received when the CRCEN bit of the SPI_CFG1 register is set. The CRCE flag of the SPI_SR register is set if the value received in the shift register does not match the receiver SPI_RXCRC value, after the last data is received (as defined by TSIZE). The CRCE flag triggers an interrupt if the CRCEIE bit is set. Clearing the bit CRCE is done by a writing 1 to the CRCEC bit of the SPI_IFCR register.

TI mode frame format error (TIFRE)

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the TIFRE flag is set in the SPI_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of a few data frames.

The TIFRE flag is cleared by writing 1 to the TIFREC bit of the SPI_IFCR register. If the TIFREIE bit is set, an interrupt is generated on the NSS error detection. As data consistency is no longer guaranteed, communication must be reinitiated by software between master and slave.

38.5.3 CRC computation

Two separate CRC calculators are implemented to check the reliability of transmitted and received data. The CRC polynomial configuration depends on the instance. Refer to *Section 38.3: SPI implementation* for more information.

The length of the polynomial is defined by the most significant bit of the value stored in the SPI_CRCPOLY register. It must be greater than the data frame size (in bits) defined in the DSIZE[4:0] bitfield of the SPI_CFG1 register. To obtain a full-size polynomial, the polynomial length must exceed the maximum data size of the peripheral instance, and the CRC33_17 bit of the SPI_CR1 register must be set to select the most significant bit of the polynomial string.

For example, to select the standard CRC16-CCITT (XMODEM) polynomial x^16 + x^12 + x^5 + 1, write 0x1021 and set the CRC33_17 bit to obtain the full size.

The CRCSIZE field of the SPI_CFG1 register then defines how many most significant bits from the CRC calculation registers are transferred and compared as CRC frame. It is defined independently from the data frame length, but it must be either equal or an integer multiple of the data frame size. Its size cannot exceed the maximum data size of the instance.

To fully benefit from the CRC calculation capability, the polynomial length setting must correspond to the CRC frame size, else the bits unused at the calculation are transferred and expected all zero at the end of the CRC frame if its size is set greater than the polynomial length.

CRC principle

The CRC calculation is enabled by setting the CRCEN bit of the SPI_CFG1 register before the SPI is enabled (SPE = 1). The CRC value is then calculated using the CRC polynomial defined by the CRCPOLY register and CRC33_17 bit. When SPI is enabled, the CRC polynomial can be changed but only in the case when there is no traffic on the bus.

The CRC computation is done, bit by bit, on the sampling clock edge defined by the CPHA and CPOL bits of the SPI_CR1 register. The calculated CRC value is checked automatically at the end of the data block defined by the SPI_CR2 register exclusively.

When a mismatch is detected between the CRC calculated internally on the received data and the CRC received from the transmitter, a CRCE flag is set to indicate a data corruption error. The right procedure for handling the CRC depends on the SPI configuration and the chosen transfer management.

CRC transfer management

Communication starts and continues normally until the last data frame has been sent or received in the SPI_DR register.

The length of the transfer must be defined by TSIZE. When the desired number of data is transferred, the TXCRC is transmitted and the data received on the line are compared to the RXCRC value.

Whatever the CRCSIZE configuration, TSIZE cannot be set to 0x3FF, if CRC is enabled.

In transmission, the CRC computation is frozen during the CRC transfer and the TXCRC is transmitted, in a frame of length equal to the CRCSIZE field value.

In reception, the RXCRC is also frozen when the desired number of data is transferred. Information to be compared with the RXCRC register content is then received in a frame of length equal to the CRCSIZE value.

Once the CRC frame is complete, an automatic check is performed comparing the received CRC value and the value calculated in the SPI_RXCRC register. The software has to check the CRCE flag of the SPI_SR register to determine if the data transfers were corrupted or not. Software clears the CRCE flag by writing 1 to the CRCEC.

The user takes no care about any flushing redundant CRC information, it is done automatically.

Resetting the SPI_TXCRC and SPI_RXCRC values

The SPI_TXCRC and SPI_RXCRC values are initialized automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode to transfer data without any interruption (several data blocks covered by intermediate CRC checking phases). Initialization patterns for receiver and transmitter can be configured either to zero or to all ones in dependency on setting bits TCRCINI and RCRCINI of the SPI_CR1 register.

The CRC values are reset when the SPI is disabled.

38.6 SPI in low-power modes

The SPI supports autonomous operation down to Stop mode, refer to *Section 38.4.15: Autonomous mode*.

Table 369. Effect of low-power modes on the SPI

Mode	Description
Sleep	No effect. SPI interrupts cause the device to exit Sleep mode.
Stop 0 and Stop 1	The content of the SPI registers is kept. If the autonomous mode is enabled at RCC configuration, and the SPI is clocked by an internal oscillator available in Stop mode, transfers are functional. The DMA requests are functional, and the interrupts in these modes cause the device to exit Stop mode.
Stop 2	The content of the SPI registers is kept. If the autonomous mode is enabled at RCC configuration, and the SPI is clocked by an internal oscillator available in Stop mode, the transfers are functional but FIFOs must be handled by software (DMA not available). The interrupts in this mode cause the device to exit Stop mode.
Stop 3	The content of the SPI registers is kept. No autonomous mode.
Standby	The SPI peripheral is powered down and must be reinitialized after exiting Standby mode.

38.7 SPI interrupts

Table 370 gives an overview of the SPI events capable of generating interrupts if enabled. Some of them feature wake-up from low-power mode capability, additionally. Most of them can be enabled and disabled independently while using specific interrupt enable control bits. The flags associated with the events are cleared by specific methods. Refer to the

description of SPI registers for more details about the event flags. When SPI is disabled, all the pending interrupt requests are blocked to prevent their propagation into the interrupt services, except the MODF interrupt request.

Table 370. SPI wake-up and interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Event clear method	Exit from Stop and Standby modes capability(1)(2)
SPI	TxFIFO ready to be loaded (space available for one data packet - FIFO threshold)	TXP	TXPIE	TXP cleared by hardware when TxFIFO contains less than FTHLV empty locations	Yes
	Data received in RxFIFO (one data packet available - FIFO threshold)	RXP	RXPIE	RXP cleared by hardware when RxFIFO contains less than FTHLV samples	Yes
	Both TXP and RXP active	DXP	DXPIE	When TXP or RXP are cleared	Yes
	Transmission Transfer Filled	TXTF	TXTFIE	Writing TXTFC to 1	No
	Underrun	UDR	UDRIE	Writing UDRC to 1	Yes
	Overrun	OVR	OVRIE	Writing OVRC to 1	Yes
	CRC Error	CRCE	CRCEIE	Writing CRCEC to 1	Yes
	TI Frame Format Error	TIFRE	TIFREIE	Writing TIFREC to 1	No
	Mode Fault	MODF	MODFIE	Writing MODFC to 1	No
	End Of Transfer (full transfer sequence completed - based on TSIZE value)	EOT	EOTIE	Writing EOTC to 1	Yes
	Master mode suspended	SUSP		Writing SUSPC to 1	Yes
	TxFIFO transmission complete (TxFIFO empty)	TXC(3)		TXC cleared by hardware when a transmission activity starts on the bus	No

1. All the interrupt events can wake up the system from Sleep mode at each instance. For detailed information about instances capabilities to exit from concrete Stop and Standby mode refer to the *Functionalities depending on the working mode* table.
2. Refer to *Section 38.3: SPI implementation* for information about Standby mode availability.
3. The TXC flag behavior depends on the TSIZE setting. When TSIZE>0, the flag fully follows the EOT one including its clearing by EOTC.

38.8 SPI registers

38.8.1 SPI control register 1 (SPI_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOLOCK
															rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
rw	rw	rw	rw	rw	w	rs	rw								rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **IOLOCK**: locking the AF configuration of associated I/Os

This bit can be changed by software only when SPI is disabled (SPE = 0). It is cleared by hardware if a MODF event occurs

When this bit is set, the SPI_CFG2 register content cannot be modified. This bit is write-protected when SPI is enabled (SPE = 1).

0: AF configuration is not locked

1: AF configuration is locked

Bit 15 **TCRCINI**: CRC calculation initialization pattern control for transmitter

0: all zero pattern is applied

1: all ones pattern is applied

Bit 14 **RCRCINI**: CRC calculation initialization pattern control for receiver

0: All zero pattern is applied

1: All ones pattern is applied

Bit 13 **CRC33_17**: Full size (17-bit) CRC polynomial configuration

0: Full size (17-bit) CRC polynomial is not used

1: Full size (17-bit) CRC polynomial is used

Bit 12 **SSI:** Internal slave select signal input level

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the peripheral NSS input internally and the I/O value of the NSS pin is ignored.

Bit 11 **HDDIR:** Rx/Tx direction at half-duplex mode

In half-duplex configuration the HDDIR bit establishes the Rx/Tx direction of the data transfer. This bit is ignored in Full-Duplex or any Simplex configuration.

0: SPI is receiver

1: SPI is transmitter

Bit 10 **CSUSP:** master suspend request

This bit reads as zero.

In master mode, when this bit is set by software, the CSTART bit is reset at the end of the current frame and communication is suspended. The user has to check SUSP flag to check end of the frame transfer.

The master mode communication must be suspended (using this bit or keeping TXDR empty) before going to Low-power mode.

After software suspension, SUSP flag must be cleared and SPI disabled and re-enabled before the next transfer starts.

Bit 9 **CSTART:** master transfer start

This bit can be set by software if SPI is enabled only to start an SPI communication. it is cleared by hardware when end of transfer (EOT) flag is set or when a transfer suspend request is accepted.

In SPI mode, the bit is taken into account at master mode only. If transmission is enabled, communication starts or continues only if any data is available in the transmission FIFO.

0: master transfer is at idle

1: master transfer is ongoing or temporary suspended by automatic suspend

Bit 8 **MASRX:** master automatic suspension in Receive mode

This bit is set and cleared by software to control continuous SPI transfer in master receiver mode and automatic management in order to avoid overrun condition.

When SPI communication is suspended by hardware automatically, it may happen that few bits of next frame are already clocked out due to internal synchronization delay.

This is why, the automatic suspension is not quite reliable when size of data drops below 8 bits. In this case, a safe suspension can be achieved by combination with delay inserted between data frames applied when MIDI parameter keeps a non zero value; sum of data size and the interleaved SPI cycles must always produce interval at length of 8 SPI clock periods at minimum. After software clearing of the SUSP bit, the communication resumes and continues by subsequent bits transfer without any next constraint. Before the SUSP bit is cleared, the user must release the RxFIFO space as much as possible by reading out all the data packets available at RxFIFO based on the RXP flag indication to prevent any subsequent suspension.

0: SPI flow/clock generation is continuous, regardless of overrun condition. (data are lost)

1: SPI flow is suspended temporary on RxFIFO full condition, before reaching overrun condition. The SUSP flag is set when the SPI communication is suspended.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **SPE:** serial peripheral enable

This bit is set by and cleared by software.

When SPE = 1, SPI data transfer is enabled, the SPI_CFG1 and SPI_CFG2 configuration registers, CRCPOLY, UDRDR, part of the SPI_AUTOCR register and IOLOCK bit in the SPI_CR1 register are write protected. They can be changed only when SPE = 0.

When SPE = 0 any SPI operation is stopped and disabled, all the pending requests of the events with enabled interrupt are blocked except the MODF interrupt request (but their pending still propagates the request of the spi_plck clock), the NSS output is deactivated at master, the RDY signal keeps not ready status at slave, the internal state machine is reseted, all the FIFOs content is flushed, CRC calculation initialized, receive data register is read zero.

SPE is cleared and cannot be set when MODF error flag is active.

0: Serial peripheral disabled.

1: Serial peripheral enabled

38.8.2 SPI control register 2 (SPI_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSIZE[15:0]**: number of data at current transfer

When these bits are changed by software, the SPI must be disabled.

Endless transfer is initialized when CSTART is set while zero value is stored in TSIZE. TSIZE cannot be set to 0x3FF, when CRC is enabled.

Note: TSIZE[15:10] bits are reserved and must be kept at reset value.

38.8.3 SPI configuration register 1 (SPI_CFG1)

Address offset: 0x008

Reset value: 0x0007 0007

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BPASS	MBR[2:0]			Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	CRCSIZE[4:0]				
rw	rw	rw	rw						rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDMAEN	RXDMAEN	Res.	Res.	Res.	Res.	UDRCFG	FTHLV[3:0]				DSIZE[4:0]				
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BPASS:** bypass of the prescaler at master baud rate clock generator

0: bypass is disabled

1: bypass is enabled

Bits 30:28 **MBR[2:0]:** master baud rate prescaler setting

000: SPI master clock/2

001: SPI master clock/4

010: SPI master clock/8

011: SPI master clock/16

100: SPI master clock/32

101: SPI master clock/64

110: SPI master clock/128

111: SPI master clock/256

Note: MBR setting is considered at slave working at TI mode, too (see Section 38.5.1: TI mode).

Bits 27:23 Reserved, must be kept at reset value.

Bit 22 **CRCEN:** hardware CRC computation enable
0: CRC calculation disabled
1: CRC calculation enabled

Bit 21 Reserved, must be kept at reset value.

Bits 20:16 **CRCSIZE[4:0]:** length of CRC frame to be transferred and compared
Most significant bits are taken into account from polynomial calculation when CRC result is transferred or compared. The length of the polynomial is not affected by this setting.
The value must be equal or a multiple of the data size configured through the DSIZE bitfield. Its maximum size corresponds to the instance maximum data size.
00111: 8 bits
01111: 16-bits
Note: The most significant bit at CRCSIZE bit field is reserved.
The CRCSIZE[2:0] bits are fixed to 1.

Bit 15 **TXDMAEN:** Tx DMA stream enable
0: Tx DMA disabled
1: Tx DMA enabled

Bit 14 **RXDMAEN:** Rx DMA stream enable
0: Rx-DMA disabled
1: Rx-DMA enabled

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **UDRCFG:** behavior of slave transmitter at underrun condition
For more details see *Figure 355: Optional configurations of the slave behavior when an underrun condition is detected*.
0: slave sends a constant pattern defined by the user in the SPI_UDRDR register
1: Slave repeats the last received data from master. When slave is configured at transmit only mode (COMM[1:0] = 01), all zeros pattern is repeated.

Bits 8:5 **FTHLV[3:0]:** FIFO threshold level
Defines number of data frames in a single data packet. It is recommended that the size of the packet does not exceed 1/2 of FIFO space.
SPI interface is more efficient if configured packet sizes are aligned with data register access parallelism:
–If SPI data register is accessed as a 16-bit register and DSIZE ≤ 8 bits, better to select FTHLV = 2, 4, 6.
–If SPI data register is accessed as a 32-bit register and DSIZE > 8 bits, better to select FTHLV = 2, 4, 6, while if DSIZE ≤ 8 bits, better to select FTHLV = 4, 8, 12.
0000: 1-data
0001: 2-data
0010: 3-data
0011: 4-data
Note: FTHLV[3:2] bits are reserved.

Bits 4:0 **DSIZE[4:0]:** number of bits in a single SPI data frame
00111: 8 bits
01111: 16-bits
Note: The most significant bit DSIZE[4] is reserved.
DSIZE[2:0] bits are fixed to 1.

38.8.4 SPI configuration register 2 (SPI_CFG2)

Address offset: 0x00C

Reset value: 0x0000 0000

The content of this register is write-protected when SPI is enabled or the IOLOCK bit is set in the SPI_CR1 register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM[1:0]		Res.
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IOSWP	RDIOP	RDIOM	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]			
rw	rw	rw						rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AFCNTR:** alternate function GPIOs control

This bit is taken into account when SPE = 0 only.

When SPI must be disabled temporary for a specific configuration reason (for example CRC reset, CPHA or HDDIR change) setting this bit prevents any glitches on the associated outputs configured at alternate function mode by keeping them forced at state corresponding the current SPI configuration.

0: The peripheral takes no control of GPIOs while it is disabled

1: The peripheral keeps always control of all associated GPIOs

Bit 30 **SSOM**: NSS output management in master mode

This bit is taken into account in master mode when SSOE is enabled. It allows the NSS output to be configured between two consecutive data transfers.

0: NSS is kept at active level until data transfer is complete, it becomes inactive with EOT flag

1: SPI data frames are interleaved with NSS nonactive pulses when MIDI[3:0] > 1

Bit 29 **SSOE:** NSS output enable

This bit is taken into account in master mode only

0: NSS output is disabled and the SPI can work in multimaster configuration

1: NSS output is enabled. The SPI cannot work in a multimaster environment. It forces the NSS pin at inactive level after the transfer is complete or SPI is disabled with respect to SSOM, MIDI, MSSI, SSIOP bits setting

Bit 28 **SSIOP**: NSS input/output polarity

0: low level is active for NSS signal

1: high level is active for NSS signal

Bit 27 Reserved, must be kept at reset value.

Bit 26 **SSM:** Software management of internal slave select signal input

0: Input value of the internal slave select signal is determined by the external NSS hardware pin

1: Input value of the internal slave select signal is determined by the SSI bit controlled by software

Note: When master uses hardware NSS output (SSM = 0 and SSOE = 1), the internal slave select signal input is forced to not active state internally to prevent master mode fault error.

Bit 25 **CPOL:** clock polarity
0: SCK signal is at 0 when idle
1: SCK signal is at 1 when idle

Bit 24 **CPHA:** clock phase
0: the first clock transition is the first data capture edge
1: the second clock transition is the first data capture edge

Bit 23 **LSBFRST:** data frame format
0: MSB transmitted first
1: LSB transmitted first

Bit 22 **MASTER:** SPI master
0: SPI slave
1: SPI master

Bits 21:19 **SP[2:0]:** serial protocol
000: SPI Motorola
001: SPI TI
others: reserved, must not be used

Bits 18:17 **COMM[1:0]:** SPI Communication Mode
00: full-duplex
01: simplex transmitter
10: simplex receiver
11: half-duplex

Bit 16 Reserved, must be kept at reset value.

Bit 15 **IOSWP:** swap functionality of MISO and MOSI pins
When this bit is set, the function of MISO and MOSI pins alternate functions are inverted. Original MISO pin becomes MOSI and original MOSI pin becomes MISO.
0: no swap
1: MOSI and MISO are swapped

Bit 14 **RDIOP:** RDY signal input/output polarity
0: high level of the signal means the slave is ready for communication
1: low level of the signal means the slave is ready for communication

Bit 13 **RDIOM:** RDY signal input/output management
0: RDY signal is defined internally fixed as permanently active (RDIOP setting has no effect)
1: RDY signal is overtaken from alternate function input (at master case) or output (at slave case) of the dedicated pin (RDIOP setting takes effect)
Note: When DSIZE in the SPI_CFG1 register is configured shorter than 8 bits, the RDIOM bit must be kept at zero.

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:4 **MIDI[3:0]**: Master Inter-Data Idleness
Specifies minimum time delay (expressed in SPI clock cycles periods) inserted between two consecutive data frames in master mode.
0000: no delay
0001: 1 clock cycle period delay
...
1111: 15 clock cycle periods delay
Note: This feature is not supported in TI mode.

Bits 3:0 **MSSI[3:0]**: Master NSS Idleness

Specifies an extra delay, expressed in number of SPI clock cycle periods, inserted additionally between active edge of NSS opening a session and the beginning of the first data frame of the session in master mode when SSOE is enabled.

0000: no extra delay

0001: 1 clock cycle period delay added

...

1111: 15 clock cycle periods delay added

Note: This feature is not supported in TI mode.
To include the delay, the SPI must be disabled and re-enabled between sessions.

38.8.5 SPI interrupt enable register (SPI_IER)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **MODFIE**: mode Fault interrupt enable
0: MODF interrupt disabled
1: MODF interrupt enabled

Bit 8 **TIFREIE**: TIFRE interrupt enable
0: TIFRE interrupt disabled
1: TIFRE interrupt enabled

Bit 7 **CRCEIE:** CRC error interrupt enable
0: CRC interrupt disabled
1: CRC interrupt enabled

Bit 6 **OVRIE:** OVR interrupt enable
0: OVR interrupt disabled
1: OVR interrupt enabled

Bit 5 **UDRIE:** UDR interrupt enable
0: UDR interrupt disabled
1: UDR interrupt enabled

Bit 4 **TXTFIE:** TXTF interrupt enable
0: TXTF interrupt disabled
1: TXTF interrupt enabled

Bit 3 **EOTIE:** EOT, SUSP and TXC interrupt enable
0: EOT/SUSP/TXC interrupt disabled
1: EOT/SUSP/TXC interrupt enabled

Bit 2 **DXPIE**: DXP interrupt enabled
DXPIE is set by software and cleared by TXTF flag set event.
0: DXP interrupt disabled
1: DXP interrupt enabled

Bit 1 **TXPIE:** TXP interrupt enable
TXPIE is set by software and cleared by TXTF flag set event.
0: TXP interrupt disabled
1: TXP interrupt enabled

Bit 0 **RXPIE:** RXP interrupt enable
0: RXP interrupt disabled
1: RXP interrupt enabled

38.8.6 SPI status register (SPI_SR)

Address offset: 0x014

Reset value: 0x0000 1002

All the flags of this register are not cleared automatically when the SPI is reenabled. They require specific clearing access exclusively via the flag clearing register as noted in the bits descriptions below.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXWNE	RXPLVL[1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DXP	TXP	RXP
r	r	r	r	r		r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **RXWNE:** RxFIFO word not empty
0: less than four bytes of RxFIFO space is occupied by data
1: at least four bytes of RxFIFO space is occupied by data
Note: This bit value does not depend on DSIZE setting and keeps together with RXPLVL[1:0] information about RxFIFO occupancy by residual data.

Bits 14:13 **RXPLVL[1:0]:** RxFIFO packing level
When RXWNE = 0, the value gives number of remaining data frames persisting at RxFIFO.
00: no next frame is available at RxFIFO
01: 1 frame is available
10: 2 frames are available*
11: 3 frames are available*
Note: (*): Possible value when data size is set up to 8 bits only.

Bit 12 **TXC:** TxFIFO transmission complete

The flag behavior depends on TSIZE setting.

When TSIZE = 0, the TXC is changed by hardware exclusively and it raises each time the TxFIFO becomes empty and there is no activity on the bus.

If TSIZE ≠ 0 there is no specific reason to monitor TXC as it just copies the EOT flag value including its software clearing. The TXC generates an interrupt when EOTIE is set.

This flag is set when SPI is reset or disabled.

0: current data transfer is still ongoing, data is available in TxFIFO or last frame transmission is on going.

1: last TxFIFO frame transmission complete

Bit 11 **SUSP:** suspension status

In master mode, SUSP is set by hardware either as soon as the current frame is complete after CSUSP request is done or at master automatic suspend receive mode (the MASRX bit is set in the SPI_CR1 register) on RxFIFO full condition.

SUSP generates an interrupt when EOTIE is set.

This bit must be cleared prior to disabling the SPI. This is done by setting the SUSPC bit of SPI_IFCR exclusively.

0: SPI not suspended (master mode active or other mode).

1: master mode is suspended (current frame completed).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODF**: mode fault

When MODF is set, SPE and IOLOCK bits of the SPI_CR1 register are reset and setting SPE again is blocked until MODF is cleared.

This bit is cleared by writing 1 to the MODFC bit of the SPI_IFCR exclusively.

0: no mode fault

1: mode fault detected.

Bit 8 **TIFRE**: TI frame format error

This bit is cleared by writing 1 to the TIFREC bit of the SPI_IFCR exclusively.

0: no TI Frame Error

1: TI frame error detected

Bit 7 **CRCE:** CRC error

This bit is cleared when SPI is re-enabled or by writing 1 to the CRCEC bit of the SPI_IFCR optionally.

0: no CRC error

1: CRC error detected

Bit 6 **OVR:** overrun

This bit is cleared when SPI is re-enabled or by writing 1 to the OVRC bit of the SPI_IFCR optionally.

0: no overrun

1: overrun detected

Bit 5 **UDR:** underrun

This bit is cleared when SPI is re-enabled or by writing 1 to the UDRC bit of the SPI_IFCR optionally.

0: no underrun

1: underrun detected

Note: In SPI mode, the UDR flag applies to slave mode only.

Bit 4 **TXTF:** transmission transfer filled

TXTF is set by hardware as soon as all of the data packets in a transfer have been submitted for transmission by application software or DMA, that is when TSIZE number of data have been pushed into the TxFIFO.

This bit is cleared by software write 1 to the TXTFC bit of the SPI_IFCR exclusively.

The TXTF flag triggers an interrupt if the TXTFIE bit is set.

TXTF setting clears the TXPIE and DXPIE masks so to off-load application software from calculating when to disable TXP and DXP interrupts.

0: upload of TxFIFO is ongoing or not started

1: TxFIFO upload is finished

Bit 3 **EOT:** end of transfer

EOT is set by hardware as soon as a full transfer is complete, that is when SPI is re-enabled or when TSIZE number of data have been transmitted and/or received on the SPI. EOT is cleared when SPI is re-enabled or by writing 1 to the EOTC bit of the SPI_IFCR register optionally.

EOT flag triggers an interrupt if the EOTIE bit is set.

If DXP flag is used until TXTF flag is set and DXPIE is cleared, EOT can be used to download the last packets contained into RxFIFO in one-shot.

In master, EOT event terminates the data transfer and handles NSS output optionally. When CRC is applied, the EOT event is extended over the CRC frame transfer.

To restart the internal state machine properly, SPI is strongly suggested to be disabled and re-enabled before next transfer starts despite its setting is not changed.

0: transfer is ongoing or not started

1: transfer complete

Bit 2 **DXP**: duplex packet

DXP flag is set whenever both TXP and RXP flags are set regardless SPI mode.

0: TxFIFO is Full and/or RxFIFO is Empty

1: both TxFIFO has space for write and RxFIFO contains for read a single packet at least

Bit 1 **TXP:** Tx-packet space available

TXP flag can be changed only by hardware. Its value depends on the physical size of the FIFO and its threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode), and actual communication flow. If the data packet is stored by performing consecutive write operations to SPI_TXDR, TXP flag must be checked again once a complete data packet is stored at TxFIFO. TXP is set despite SPI TxFIFO becomes inaccessible when SPI is reset or disabled.

0: not enough free space at TxFIFO to host next data packet

1: enough free space at TxFIFO to host at least one data packet

Bit 0 **RXP:** Rx-packet available

The flag is changed by hardware. It monitors the total number of data currently available at RxFIFO if SPI is enabled. RXP value depends on the FIFO threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode), and actual communication flow. If the data packet is read by performing consecutive read operations from SPI_RXDR, RXP flag must be checked again once a complete data packet is read out from RxFIFO.

0: RxFIFO is empty or an incomplete data packet is received

1: RxFIFO contains at least one data packet

38.8.7 SPI interrupt/status flags clear register (SPI_IFCR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SUSPC	Res.	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
				w		w	w	w	w	w	w	w			

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SUSPC:** Suspend flag clear
Writing a 1 into this bit clears SUSP flag of the SPI_SR register

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODFC**: mode fault flag clear
Writing a 1 into this bit clears MODF flag of the SPI_SR register

Bit 8 **TIFREC**: TI frame format error flag clear
Writing a 1 into this bit clears TIFRE flag of the SPI_SR register

Bit 7 **CRCEC:** CRC error flag clear
Writing a 1 into this bit clears CRCE flag of the SPI_SR register

Bit 6 **OVRC:** overrun flag clear
Writing a 1 into this bit clears OVR flag of the SPI_SR register

Bit 5 **UDRC:** underrun flag clear
Writing a 1 into this bit clears UDR flag of the SPI_SR register

Bit 4 **TXTFC:** transmission transfer filled flag clear
Writing a 1 into this bit clears TXTF flag of the SPI_SR register

Bit 3 **EOTC:** end of transfer flag clear
Writing a 1 into this bit clears EOT flag of the SPI_SR register

Bits 2:0 Reserved, must be kept at reset value.

38.8.8 SPI autonomous mode control register (SPI_AUTOCR)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGEN	TRIGPOL	TRIGSEL[3:0]			
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **TRIGEN:** Hardware control of CSTART triggering enable
0: Hardware control disabled
1: Hardware control enabled
Note: if user cannot prevent trigger event during write, the TRIGEN must be changed when SPI is disabled

Bit 20 **TRIGPOL:** Trigger polarity
0: trigger is active on raising edge
1: trigger is active on falling edge
Note: This bit can be written only when SPE = 0.

Bits 19:16 **TRIGSEL[3:0]**: Trigger selection (refer *Description of SPI interconnections*).
0000: spi_trg0 is selected
0001: spi_trg1 is selected
...
1111: spi_trg15 is selected
Note: these bits can be written only when SPE = 0.

Bits 15:0 Reserved, must be kept at reset value.

38.8.9 SPI transmit data register (SPI_TXDR)

Address offset: 0x020

Reset value: 0x0000 0000

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

Bits 31:0 **TXDR[31:0]:** transmit data register
The register serves as an interface with TxFIFO. A write to it accesses TxFIFO.
Note: data is always right-aligned. Unused bits are ignored when writing to the register, and read as zero when the register is read.
Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is written by single access.
half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word-data can be written by single access.
word-wise (32 bit access). In this case 4 data-bytes or 2 half-word-data or word-data can be written by single access.
Write access of this register less than the configured data size is forbidden.

38.8.10 SPI receive data register (SPI_RXDR)

Address offset: 0x030

Reset value: 0x0000 0000

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

Bits 31:0 **RXDR[31:0]:** receive data register

The register serves as an interface with RxFIFO. When it is read, RxFIFO is accessed.

Note: data is always right-aligned. Unused bits are read as zero when the register is read. Writing to the register is ignored.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is read by single access

half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word data can be read by single access

word-wise (32 bit access). In this case 4 data-bytes or 2 half-word data or word-data can be read by single access.

Read access of this register less than the configured data size is forbidden.

38.8.11 SPI polynomial register (SPI_CRCPOLY)

Address offset: 0x040

Reset value: 0x0000 0107

The content of this register is write protected when SPI is enabled.

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

Bits 31:0 **CRCPOLY[31:0]:** CRC polynomial register

This register contains the polynomial for the CRC calculation.

The default 9-bit polynomial setting 0x107 corresponds to default 8-bit setting of DSIZE. It is compatible with setting 0x07 used in other ST products with fixed length of the polynomial string, where the most significant bit of the string is always kept hidden.

Length of the polynomial is given by the most significant bit of the value stored in this register. It must be set greater than DSIZE. CRC33_17 bit must be set additionally with CRCPOLY register when DSIZE is configured to maximum data size and CRC is enabled (to keep polynomial length grater than data size).

Note: CRCPOLY[31:16] bits are reserved. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

38.8.12 SPI transmitter CRC register (SPI_TXCRC)

Address offset: 0x044

Reset value: 0x0000 0000

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

Bits 31:0 **TXCRC[31:0]:** CRC register for transmitter

When CRC calculation is enabled, the TXCRC[31:0] bits contain the computed CRC value of the subsequently transmitted bytes. CRC calculation is initialized when the CRCEN bit of the SPI_CR1 register is set or when a data block is transferred completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on the SPI_CRCPOLY register and CRCSIZE bits settings in the SPI_CFG1 register.

Note: A read to this register when the communication is ongoing may return an incorrect value.

Note: TXCRC[31-16] bits are reserved. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of the CRCSIZE bitfield is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.

38.8.13 SPI receiver CRC register (SPI_RXCRC)

Address offset: 0x048

Reset value: 0x0000 0000

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

Bits 31:0 **RXCRC[31:0]:** CRC register for receiver

When CRC calculation is enabled, the RXCRC[31:0] bits contain the computed CRC value of the subsequently received bytes. CRC calculation is initialized when the CRCEN bit of the SPI_CR1 register is set or when a data block is transferred completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.
The number of bits considered at calculation depends on the SPI_CRCPOLY register and CRCSIZE bits settings in the SPI_CFG1 register.

Note: A read to this register when the communication is ongoing may return an incorrect value.

Note: RXCRC[31-16] bits are reserved. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of CRCSIZE bit field is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.

38.8.14 SPI underrun data register (SPI_UDRDR)

Address offset: 0x04C

Reset value: 0x0000 0000

The content of this register is write protected when SPI is enabled.

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

Bits 31:0 **UDRDR[31:0]:** data at slave underrun condition

The register is taken into account in slave mode and at underrun condition only. The number of bits considered depends on the DSIZE bit setting in the SPI_CFG1 register. Underrun condition handling depends on the UDRCFG bit setting in the SPI_CFG1 register.

Note: UDRDR[31-16] bits are reserved. There is no constraint when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

38.8.15 SPI register map

Table 371. SPI register map and reset values

Offset	Register name	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
0x000	SPI_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOLOCK	TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
	Reset value																0	0	0	0	0	0	0	0	0								0
0x004	SPI_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSIZE[15:0](1)															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	SPI_CFG1	BPASS	MBR[2:0]			Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	CRCSIZE[4:0](1)					TXDMAEN	RXDMAEN	Res.	Res.	Res.	Res.	UDRCFG	FTHLV[3:0](1)				DSIZE[4:0](1)				
	Reset value	0	0	0	0						0		0	0	1	1	1	0	0					0	0	0	0	0	0	0	1	1	1
0x00C	SPI_CFG2	AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM[1:0]		Res.	IOSWP	RDIOP	RDIOM	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]			
	Reset value	0	0	0	0		0	0	0	0	0	0	0	0	0	0		0	0	0						0	0	0	0	0	0	0	0
0x010	SPI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x014	SPI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXWNE	RXPLVL[1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DPXP	TXP	RXP
	Reset value																	0	0	0	1	0		0	0	0	0	0	0	0	0	1	0
0x018	SPI_IFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSPC	Res.	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
	Reset value																					0		0	0	0	0	0	0	0			
0x01C	SPI_AUTOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGEN	TRIGPOL	TRIGSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0	0	0	0	0	0																
0x020	SPI_TXDR	TXDR[31:16]																TXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024-0x02C	Reserved	Res.																															
0x030	SPI_RXDR	RXDR[31:16]																RXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034 - 0x03C	Reserved	Res.																															
0x040	SPI_CRCPOLY	CRCPOLY[31:16](2)																CRCPOLY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1
0x044	SPI_TXCRC	TXCRC[31:16](2)																TXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	SPI_RXCRC	RXCRC[31:16](2)																RXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	SPI_UDRDR	UDRDR[31:16](2)																UDRDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. The configuration of this bitfield depends on the features of the SPI instance. For more details, refer to *Section 38.3: SPI implementation*.
2. The bits 31-16 are reserved. There is no constraint when the 32-bit access is applied at these addresses. The bits 31-16, when reserved, are always read to zero while any write to them is ignored.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

39 Serial audio interface (SAI)

39.1 SAI introduction

The SAI interface (serial audio interface) offers a wide set of audio protocols due to its flexibility and wide range of configurations. Many audio protocols can be addressed thanks to SAI “free protocol mode”. The free protocol mode allows the application to define the audio frame shape, number of slots, slot size, data size, and so on. For example, the SAI supports I2S standards, LSB- or MSB-justified, PCM/DSP, TDM, and AC’97 protocols. SPDIF output is offered when the audio block is configured as a transmitter.

To bring this level of flexibility and reconfigurability, the SAI contains two independent audio subblocks. Each block has it own clock generator and I/O line controller.

The SAI works in master or slave configuration. The audio subblocks are either receiver or transmitter and work synchronously or not (with respect to the other one).

39.2 SAI main features

- Two independent audio subblocks, which can be transmitters or receivers with their respective FIFO
- 8-word integrated FIFOs for each audio subblock
- Synchronous or asynchronous mode between the audio subblocks
- Master or slave configuration independent for both audio subblocks
- Clock generator for each audio block to target independent audio frequency sampling when both audio subblocks are configured in master mode
- Data size configurable: 8-, 10-, 16-, 20-, 24-, 32-bit
- Audio protocol: I2S, LSB- or MSB-justified, PCM/DSP, TDM, AC’97
- PDM interface, supporting up to 4 microphone pairs
- SPDIF output available if required
- Up to 16 slots available with configurable size
- Number of bits by frame can be configurable
- Frame synchronization active level configurable (offset, bit length, level)
- First active bit position in the slot is configurable
- LSB first or MSB first for data transfer
- Mute mode
- Stereo/Mono audio frame capability
- Communication clock strobing edge configurable (SCK)
- Error flags with associated interrupts if enabled respectively
  - Overrun and underrun detection
  - Anticipated frame synchronization signal detection in slave mode
  - Late frame synchronization signal detection in slave mode
  - Codec not ready for the AC’97 mode in reception
- Interrupt sources when enabled:
  - Errors

  - FIFO requests
- 2-channel DMA interface

39.3 SAI implementation

Table 372. SAI features (1)(2)

SAI features	SAI1
I2S, LSB or MSB-justified, PCM/DSP, TDM, AC'97	X
FIFO size	8 words
SPDIF	X
PDM	X(3)

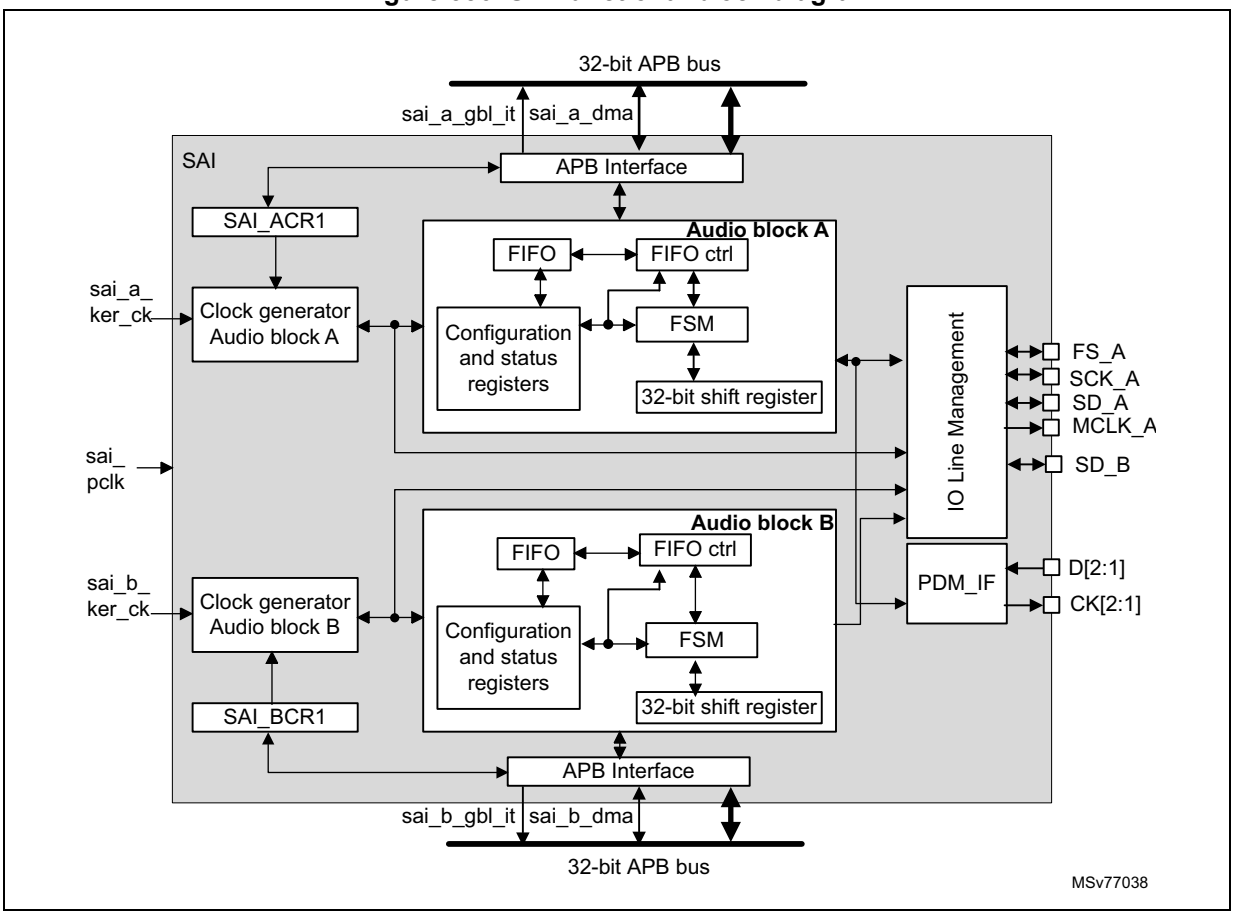
1. 'X' = supported, '-' = not supported.
2. Only SAI_FS_A, SAI_MCLK_A, SAI_FS_A, SAI_SD_A and SAI_SD_B are available.
3. Only signals D[2:1], and CK[2:1] are available.

39.4 SAI functional description

39.4.1 SAI block diagram

Figure 356 shows the SAI block diagram while *Table 373* and *Table 374* list SAI internal and external signals.

Figure 356. SAI functional block diagram



1. Refer to *Table 374* and section *Section 39.3: SAI implementation* to check which IOs are available in this product.

The SAI is mainly composed of two audio subblocks with their own clock generator. Each audio block integrates a 32-bit shift register controlled by their own functional state machine. Data are stored or read from the dedicated FIFO. FIFO may be accessed by the CPU, or by DMA to leave the CPU free during the communication. Each audio block is independent. They can be synchronous with each other.

An I/O line controller manages a set of 4 dedicated pins (SD, SCK, FS, MCLK) for a given audio block in the SAI. Some of these pins can be shared if the two subblocks are declared as synchronous to leave some free to be used as general purpose I/Os. The MCLK pin can be output, or not, depending on the application, the decoder requirement and whether the audio block is configured as the master.

If one SAI is configured to operate synchronously with another one, even more I/Os can be freed (except for pins SD_x).

The functional state machine can be configured to address a wide range of audio protocols. Some registers are present to set up the desired protocols (audio frame waveform generator).

The audio subblock can be a transmitter or receiver, in master or slave mode. The master mode means the SCK_x bit clock and the frame synchronization signal are generated from the SAI, whereas in slave mode, they come from another external or internal master. There is a particular case for which the FS signal direction is not directly linked to the master or slave mode definition. In AC'97 protocol, it is an SAI output even if the SAI (link controller) is set up to consume the SCK clock (and so to be in Slave mode).

Note: For ease of reading of this section, the notation SAI_x refers to SAI_A or SAI_B, where 'x' represents the SAI A or B subblock.

39.4.2 SAI pins and internal signals

Table 373. SAI internal input/output signals

Internal signal name	Signal type	Description
sai_a_gbl_it/ sai_b_gbl_it	Output	Audio block A and B global interrupts
sai_a_dma, sai_b_dma	Input/output	Audio block A and B DMA acknowledges and requests
sai_a_ker_ck/ sai_b_ker_ck	Input	Audio block A/B kernel clock
sai_pclk	Input	APB clock

Table 374. SAI input/output pins

Pin name	Pin type	Comments
SAI_SCK_A	Input/output	Audio block A bit clock
SAI_MCLK_A	Output	Audio block A master clock
SAI_FS_A	Input/output	Frame synchronization line for audio block A
SAI_SD_A	Input/output	Data line for block A
SAI_SD_B	Input/output	Data line for block B
SAI_CK[2:1]	Output	PDM bitstream clock
SAI_D[2:1]	Input	PDM bitstream data

39.4.3 Main SAI modes

Each audio subblock of the SAI can be configured to be master or slave via the MODE bits in the SAI_xCR1 register of the selected audio block.

Master mode

In master mode, the SAI delivers the timing signals to the external connected device:

- The bit clock and the frame synchronization are output on pin SCK_x and FS_x, respectively.
- If needed, the SAI can also generate a master clock on the MCLK_x pin.

Both SCK_x, FS_x and MCLK_x are configured as outputs.

Slave mode

The SAI expects to receive timing signals from an external device.

- If the SAI subblock is configured in asynchronous mode, then the SCK_x and FS_x pins are configured as inputs.
- If the SAI subblock is configured to operate synchronously with the second audio subblock, the corresponding SCK_x and FS_x pins are left free to be used as general purpose I/Os.

In slave mode, the MCLK_x pin is not used and can be assigned to another function.

It is recommended to enable the slave device before enabling the master.

Configuring and enabling SAI modes

Each audio block can use a different audio protocol. When PRTCFG[1:0] of the SAI_xCR1 register is set to 0, the free protocol mode is selected and each SAI subblock can emulate I2S standards, LSB- or MSB-justified, PCM/DSP, TDM, or AC'97 protocols.

Each audio subblock can be independently defined as a transmitter or receiver through the MODE bit in the SAI_xCR1 register of the corresponding audio block. As a result, the SAI_SD_x pin is respectively configured as an output or an input.

Two master audio blocks in the same SAI can be configured with two different MCLK and SCK clock frequencies. In this case, they have to be configured in asynchronous mode.

Each of the audio blocks in the SAI is enabled by the SAIEN bit in the SAI_xCR1 register. As soon as this bit is active, the transmitter or the receiver is sensitive to the activity on the clock, data, and synchronization lines in slave mode.

In master Tx mode, enabling the audio block immediately generates the bit clock for the external slaves even if there is no data in the FIFO. However, FS signal generation is conditioned by the presence of data in the FIFO. After the FIFO receives the first data to transmit, this data is output to external slaves. If there is no data to transmit in the FIFO, 0 values are then sent in the audio frame with an underrun flag generation.

In slave mode, the audio frame starts when the audio block is enabled and when a start of frame is detected.

In Slave Tx mode, no underrun event is possible on the first frame after the audio block is enabled, because the mandatory operating sequence in this case is:

1. Write into the SAI_xDR (by software or by DMA).
2. Wait until the FIFO threshold *(*FLH) flag is different from 0b000 (FIFO empty).
3. Enable the audio block in slave transmitter mode.

39.4.4 SAI synchronization mode

SAI subclock A and B can be synchronized.

Internal synchronization

An audio subblock can be configured to operate synchronously with the second audio subblock in the same SAI. In this case, the bit clock and the frame synchronization signals are shared to reduce the number of external pins used for the communication. The audio block configured in synchronous mode sees its own SCK_x, FS_x, and MCLK_x pins

released back as GPIOs while the audio block configured in asynchronous mode is the one for which FS_x and SCK_x ad MCLK_x I/O pins are relevant (if the audio block is considered as master).

Typically, the audio block in synchronous mode can be used to configure the SAI in full duplex mode. One of the two audio blocks can be configured as a master and the other as slave, or both as slaves with one asynchronous block (corresponding SYNCEN[1:0] bits set to 00 in SAI_xCR1) and one synchronous block (corresponding SYNCEN[1:0] bits set to 01 in the SAI_xCR1 register).

Note: Due to internal resynchronization stages, PCLK APB frequency must be higher than twice the bit rate clock frequency.

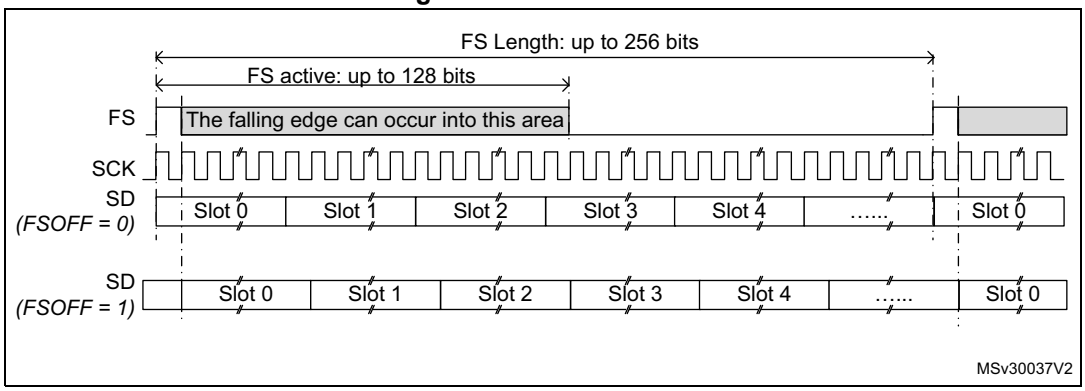
39.4.5 Audio data size

The audio frame can target different data sizes by configuring bit DS[2:0] in the SAI_xCR1 register. The data sizes may be 8, 10, 16, 20, 24, or 32 bits. During the transfer, either the MSB or the LSB of the data is sent first, depending on the configuration of the LSBFIRST bit in the SAI_xCR1 register.

39.4.6 Frame synchronization

The FS signal acts as the frame synchronization signal in the audio frame (start of frame). The shape of this signal is completely configurable to target the different audio protocols with their own specificities concerning this frame synchronization behavior. This reconfigurability is done using the SAI_xFRCR register. *Figure 357* illustrates this flexibility.

Figure 357. Audio frame



In AC'97 mode or in SPDIF mode (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01 in the SAI_xCR1 register), the frame synchronization shape is forced to match the AC'97 protocol. The SAI_xFRCR register value is ignored.

Each audio block is independent and consequently each one requires a specific configuration.

Frame length

- Master mode

  The audio frame length can be configured to up to 256-bit clock cycles, by configuring the FRL[7:0] field in the SAI_xFRCR register.

  If the frame length is greater than the number of declared slots for the frame, the remaining bits to transmit are extended to 0 or the SD line is released to high-Z

depending on the state of bit TRIS in the SAI_xCR2 register (refer to *FS signal role*). In reception mode, the remaining bit is ignored.

If bit NODIV is cleared, (FRL+1) must be equal to a power of 2, from 8 to 256, to ensure that an audio frame contains an integer number of MCLK pulses per bit clock cycle.

If bit NODIV is set, the (FRL+1) field can take any value from 8 to 256. Refer to *Section 39.4.8: SAI clock generator*"

- Slave mode

  The audio frame length is mainly used to specify to the slave the number of bit clock cycles per audio frame sent by the external master. It is used mainly to detect from the master any anticipated or late occurrence of the frame synchronization signal during an ongoing audio frame. In this case, an error is generated. For more details, refer to *Section 39.4.14: Error flags*.

  In slave mode, there are no constraints on the FRL[7:0] configuration in the SAI_xFRCR register.

The number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame is 8.

Frame synchronization polarity

The FSPOL bit in the SAI_xFRCR register sets the active polarity of the FS pin from which a frame is started. The start of the frame is edge sensitive.

In slave mode, the audio block waits for a valid frame to start transmitting or receiving. The start of the frame is synchronized to this signal. It is effective only if the start of the frame is not detected during an ongoing communication and assimilated to an anticipated start of frame (refer to *Section 39.4.14: Error flags*).

In master mode, the frame synchronization is sent continuously each time an audio frame is complete until the SAIEN bit in the SAI_xCR1 register is cleared. If no data are present in the FIFO at the end of the previous audio frame, an underrun condition is managed as described in *Section 39.4.14: Error flags*), but the audio communication flow is not interrupted.

Frame synchronization active level length

The FSALL[6:0] bits of the SAI_xFRCR register enable the configuration of the length of the active level of the frame synchronization signal. The length can be set from 1- to 128-bit clock cycles.

As an example, the active length can be half of the frame length in I2S, LSB or MSB-justified modes, or one-bit wide for PCM/DSP or TDM.

Frame synchronization offset

Depending on the audio protocol targeted in the application, the frame synchronization signal can be asserted when transmitting the last bit or the first bit of the audio frame (this is the case in I2S standard protocol and in MSB-justified protocol, respectively). The FSOFF bit in the SAI_xFRCR register enables the possibility to choose between two configurations.

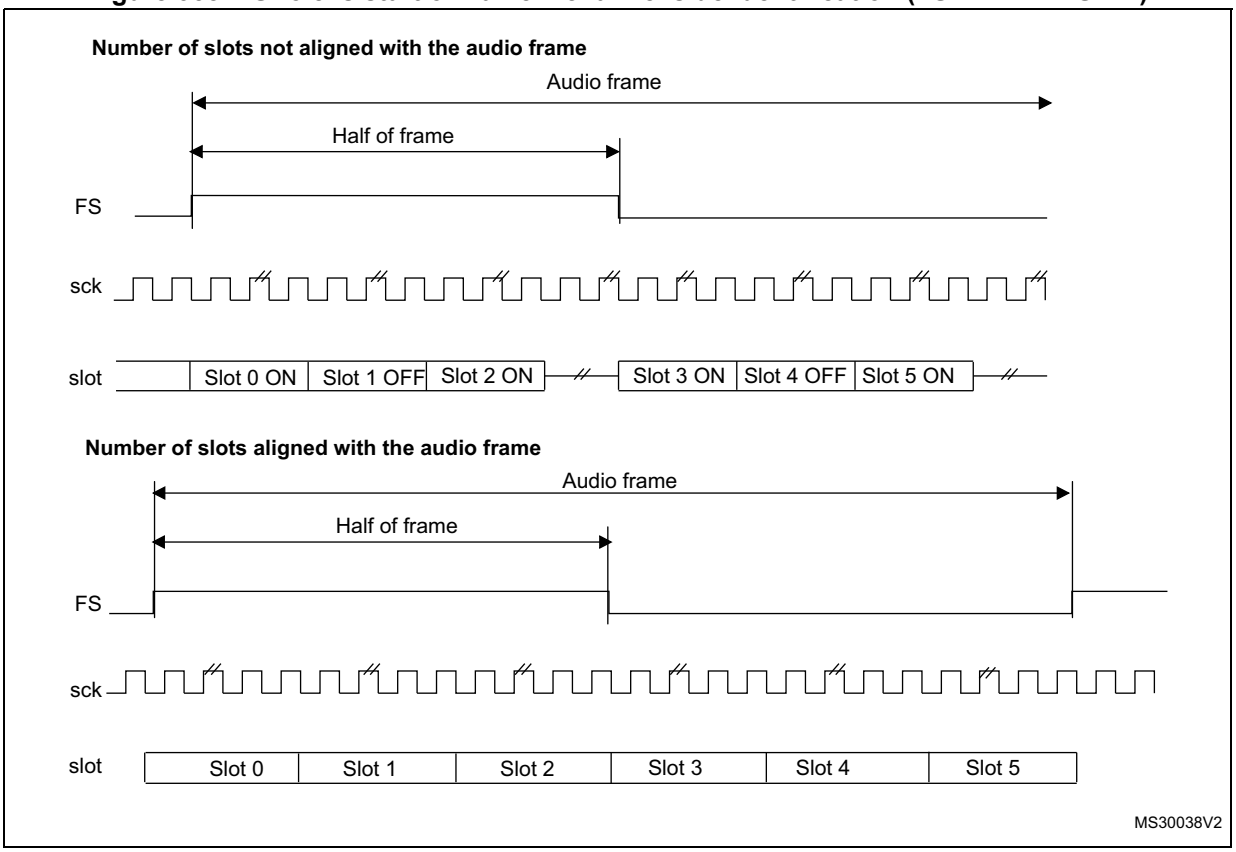
FS signal role

The FS signal can have a different meaning depending on the FS function. FSDEF bit in the SAI_xFRCR register selects which meaning it has:

- 0: start of frame, like, for instance, the PCM/DSP, TDM, AC'97, audio protocols,
- 1: start of frame and channel side identification within the audio frame like for the I2S or the MSB- or LSB-justified protocols.

When the FS signal is considered as a start of frame and channel side identification within the frame, the number of declared slots must be considered to be half the number for the left channel and half the number for the right channel. If the number of bit clock cycles on half audio frame is greater than the number of slots dedicated to a channel side, and TRIS = 0, 0 is sent for transmission for the remaining bit clock cycles in the SAI_xCR2 register. Otherwise, if TRIS = 1, the SD line is released to high-Z. In reception mode, the remaining bit clock cycles are not considered until the channel side changes.

Figure 358. FS role is start of frame + channel side identification (FSDEF = TRIS = 1)

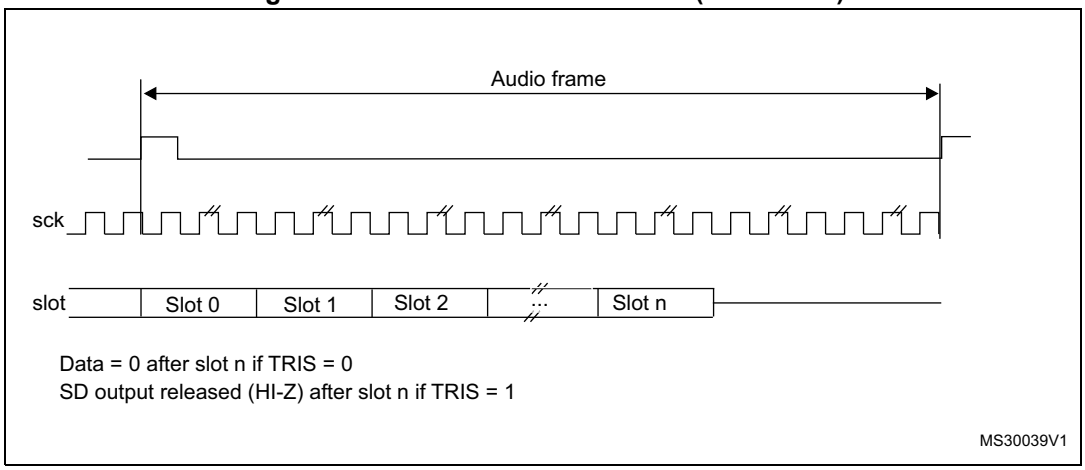


1. The frame length must be even.

If the FSDEF bit in SAI_xFRCR is kept clear, so FS signal is equivalent to a start of frame, and if the number of slots defined in NBSLOT[3:0] in SAI_xSLOTR multiplied by the number of bits by slot configured in SLOTSZ[1:0] in SAI_xSLOTR is less than the frame size (bit FRL[7:0] in the SAI_xFRCR register), then:

- If TRIS = 0 in the SAI_xCR2 register, the remaining bit after the last slot is forced to 0 until the end of frame in case of transmitter,
- If TRIS = 1, the line is released to high-Z during the transfer of these remaining bits. In reception mode, these bits are discarded.

Figure 359. FS role is start of frame (FSDEF = 0)



The FS signal is not used when the audio block in transmitter mode is configured to get the SPDIF output on the SD line. The corresponding FS I/O is released and left free for other purposes.

39.4.7 Slot configuration

The slot is the basic element in the audio frame. The number of slots in the audio frame is equal to NBSLOT[3:0] + 1.

The maximum number of slots per audio frame is fixed at 16.

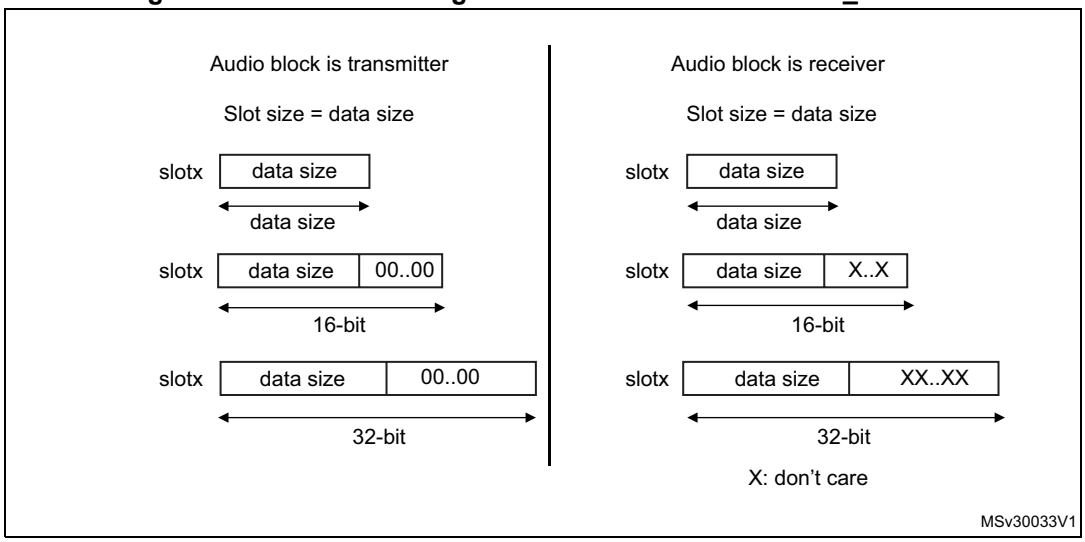
For AC'97 protocol or SPDIF (when bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the number of slots is automatically set to target the protocol specification, and the value of NBSLOT[3:0] is ignored.

Each slot can be defined as a valid slot, or not, by setting SLOTEN[15:0] bits of the SAI_xSLOTR register.

When an invalid slot is transferred, the SD data line is either forced to 0 or released to high-Z depending on the TRIS bit configuration (refer to *Output data line management on an inactive slot*) in transmitter mode. In receiver mode, the received value from the end of this slot is ignored. Consequently, there is no FIFO access and so no request to read or write the FIFO linked to this inactive slot status.

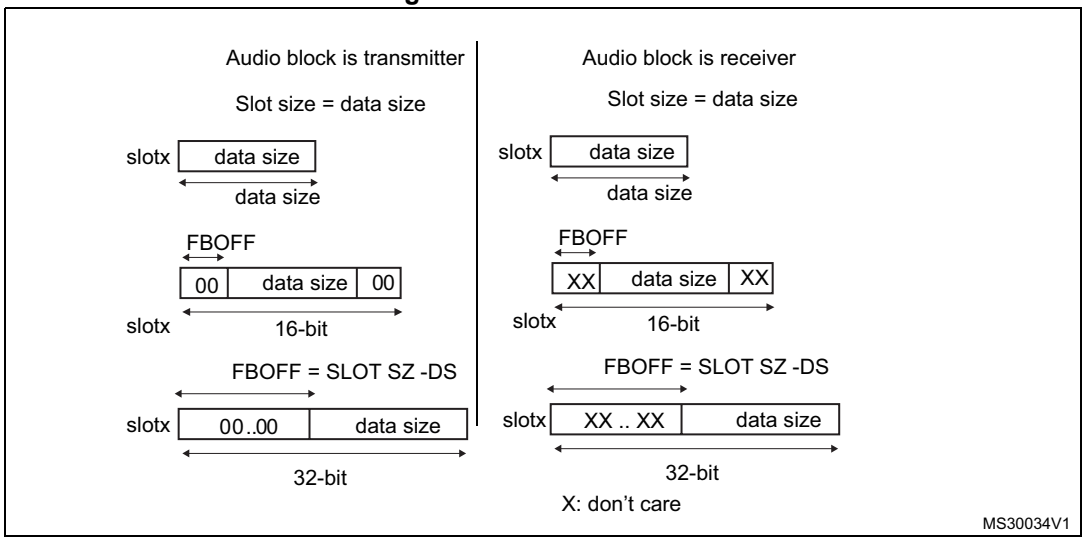
The slot size is also configurable as shown in *Figure 360.* The size of the slots is selected by configuring the SLOTSZ[1:0] bits in the SAI_xSLOTR register. The size is applied identically for each slot in an audio frame.

Figure 360. Slot size configuration with FBOFF = 0 in SAI_xSLOTR



It is possible to choose the position of the first data bit to transfer within the slots. This offset is configured by FBOFF[4:0] bits in the SAI_xSLOTR register. 0 values are injected in transmitter mode from the beginning of the slot until this offset position is reached. In reception, the bit in the offset phase is ignored. This feature targets the LSB justified protocol (if the offset is equal to the slot size minus the data size).

Figure 361. First bit offset



It is mandatory to respect the following conditions to avoid bad SAI behavior:

- FBOFF ≤ (SLOTSZ - DS)
- DS ≤ SLOTSZ
- NBSLOT x SLOTSZ ≤ FRL (frame length)

The number of slots must be even when bit FSDEF in the SAI_xFRCR register is set.

In AC'97 and SPDIF protocol (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the slot size is automatically set as defined in *Section 39.4.11: AC'97 link controller*.

39.4.8 SAI clock generator

Each audio block has its own clock generator. The clock generator builds the master clock (MCLK_x) and bit clock (SCK_x) signals from the sai_x_ker_ck. The sai_x_ker_ck clock is delivered by the clock controller of the product (RCC).

Generation of the master clock (MCLK_x)

The clock generator provides the master clock (MCLK_x) when the audio block is defined as Master or Slave. The master clock is generated as soon as the MCKEN bit is set to 1 even if the SAIEN bit for the corresponding block is set to 0. This feature can be useful if the MCLK_x clock is used as system clock for an external audio device, since it enables the generation of the MCLK_x before activating the audio stream.

To generate a master clock on MCLK_x output before transferring the audio samples, the user application has to follow the sequence below:

1. Check that SAIEN = 0.
2. Program the MCKDIV[5:0] divider to the required value.
3. Set the MCKEN bit to 1.
4. Later, the application can configure other parts of the SAI, and sets the SAIEN bit to 1 to start the transfer of audio samples.

To avoid disturbances on the clock generated on MCLK_x output, the following operations are not recommended:

- Changing MCKDIV when MCKEN = 1
- Setting MCKEN to 0 if the SAIEN = 1

The SAI makes sure that there are no spurs on MCLK_x output when the MCLK_x is switched ON and OFF via the MCKEN bit (with SAIEN = 0).

Table 375 shows MCLK_x activation conditions.

Table 375. MCLK_x activation conditions

MCLKEN	NODIV	SAIEN for block x	MCLK_x
0	X	0	Disabled
1			Enabled
0	1	1	Disabled
1			Enabled
X	0		Enabled

Note: MCLK_x can also be generated in AC'97 mode, when MCLKEN is set to 1.

Generation of the bit clock (SCK_x)

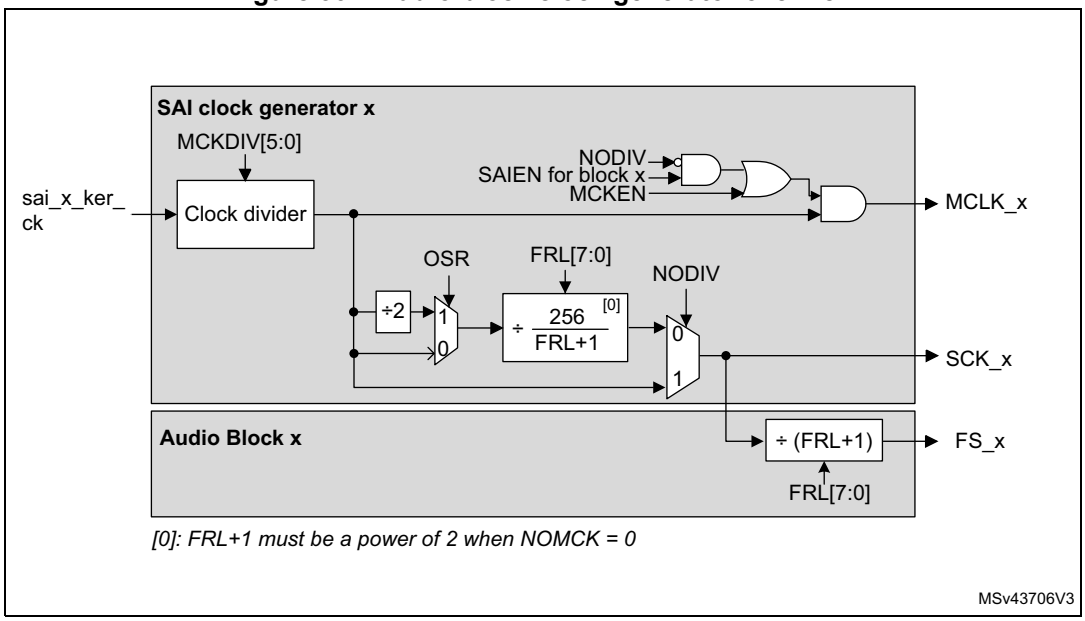
The clock generator provides the bit clock (SCK_x) when the audio block is defined as Master. The frame synchronization (FS_x) is also derived from the signals provided by the clock generator.

In Slave mode, the value of NODIV and OSR fields are ignored, and the SCK_x clock is not generated.

The bit clock strobing edge of SCK_x can be configured through the CKSTR fields, which is functional both in master and slave mode.

Figure 362 illustrates the architecture of the audio block clock generator.

Figure 362. Audio block clock generator overview



The NODIV bit must be used to force the ratio between the master clock (MCLK_x) and the frame synchronization (FS_x) frequency to 256 or 512.

- If NODIV is set to 0, the frequency ratio between the frame synchronization and the master clock is fixed to 512 or 256, according to OSR value, but the frame length must be a power of 2. More details are given below.
- If NODIV is set to 1, the application can adjust the frequency of the bit clock (SCK_x) via MCKDIV. In addition, there is no restriction on the frame length value as long as the frame length is bigger or equal to 8 (that is, FRL[7:0] > 6). The frame synchronization frequency depends on MCKDIV and frame length (FRL[7:0]). In that case, the frequency of the MCLK_x is equal to the SCK_x.

The NODIV, MCKEN, SAIEN, OVR, CKSTR, and MCKDIV[5:0] bits belong to the SAI_xCR1 register, while FRL[7:0] belongs to SAI_xFRCR.

Clock generator programming when NODIV = 0

In that case, the MCLK_x frequency is:

- $F_{MCLK_x} = 256 \times F_{FS_x}$ if OSR = 0
- $F_{MCLK_x} = 512 \times F_{FS_x}$ if OSR = 1

When MCKDIV is different from 0, the MCLK_x frequency is given by the formula below:

$$F_{MCLK_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV}$$

The frame synchronization frequency is given by:

$$F_{FS_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV \times (OSR + 1) \times 256}$$

The bit clock frequency (SCK_x) is given by the following formula:

$$F_{SCK_x} = \frac{F_{sai_x_ker_ck} \times (FRL + 1)}{MCKDIV \times (OSR + 1) \times 256}$$

Note: When NODIV is equal to 0, (FRL+1) must be a power of two. In addition, (FRL+1) must range between 8 and 256. (FRL +1) represents the number of bit clock in the audio frame.

When the MCKDIV division ratio is odd, the MCLK duty cycle is not 50%. The bit clock signal (SCK_x) can also have a duty cycle different from 50% if MCKDIV is odd, if OSR is equal to 0, and if (FRL+1) = 2^8.

It is recommended, to program MCKDIV to an even value or to large values (higher than 10).

Note that MCKDIV = 0 gives the same result as MCKDIV = 1.

Clock generator programming when NODIV = 1

When MCKDIV is different from 0, the frequency of the bit clock (SCK_x) is given in the formula below:

$$F_{SCK_x} = F_{MCLK_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV}$$

The frequency of the frame synchronization (FS_x) in given by the following formula:

$$F_{FS_x} = \frac{F_{sai_x_ker_ck}}{(FRL + 1) \times MCKDIV}$$

Note: When NODIV is set to 1, (FRL+1) can take any values from 8 to 256.

MCKDIV = 0 gives the same result as MCKDIV = 1.

Clock generator programming examples

Table 376 gives programming examples for 48, 96, and 192 kHz.

Table 376. Clock generator programming examples

Input sai_x_ker_ck clock frequency	MCLK	F_{MCLK}/ F_{FS}	FRL (1)	OSR	NODIV	MCKEN	MCKDIV[5:0]	Audio sampling frequency (F_{FS})
98.304 MHz	Y	512	2^N-1	1	0	1	0 or 1	192 kHz
		512	2^N-1	1	0	1	2	96 kHz
		512	2^N-1	1	0	1	4	48 kHz
		256	2^N-1	0	0	1	2	192 kHz
		256	2^N-1	0	0	1	4	96 kHz
		256	2^N-1	0	0	1	8	48 kHz
	N	-	63	-	1	0	8	192 kHz
		-	63	-	1	0	16	96 kHz
		-	63	-	1	0	32	48 kHz

1. N is an integer value between 3 and 8.

39.4.9 Internal FIFOs

Each audio block in the SAI has its own FIFO. Depending on if the block is defined to be a transmitter or a receiver, the FIFO can be written or read, respectively. Thus, there is only one FIFO request linked to the FREQ bit in the SAI_xSR register.

An interrupt is generated if the FREQIE bit is enabled in the SAI_xIM register. This depends on:

- The FIFO threshold setting (FLVL bits in SAI_xCR2).
- Communication direction (transmitter or receiver). Refer to *Interrupt generation in transmitter mode* and *Interrupt generation in reception mode*.

Interrupt generation in transmitter mode

The interrupt generation depends on the FIFO configuration in transmitter mode:

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if no data are available in the SAI_xDR register (FLVL[2:0] bits in SAI_xSR are less than 0b001). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is no longer empty (FLVL[2:0] bits in SAI_xSR are different from 0b000), that is, one or more data are stored in the FIFO.
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 0b001), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if less than a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b010). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b010).

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if less than half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b011). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b011).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO three quarter (FTH[2:0] set to 0b011), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if less than three quarters of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b100). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least three quarters of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b100).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if the FIFO is not full (FLVL[2:0] bits in SAI_xSR are less than 0b101). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is full (FLVL[2:0] bits in SAI_xSR are equal to 0b101).

Interrupt generation in reception mode

The interrupt generation depends on the FIFO configuration in reception mode:

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least one data is available in the SAI_xDR register (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b001). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO becomes empty (FLVL[2:0] bits in the SAI_xSR are equal to 0b000), that is, no data are stored in FIFO.
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 0b001), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least one quarter of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b010). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when less than a quarter of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR are less than 0b010).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least half of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b011). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when less than half of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR are less than 0b011).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO three quarter full (FTH[2:0] set to 0b011), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least three quarters of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b100). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO has less than three quarters of the FIFO data locations available (FLVL[2:0] bits in SAI_xSR is less than 0b100).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if the FIFO is full (FLVL[2:0] bits in SAI_xSR are equal to 0b101). This

interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is not full (FLVL[2:0] bits in SAI_xSR are less than 0b101).

Like interrupt generation, the SAI can use the DMA if the DMAEN bit in the SAI_xCR1 register is set. The FREQ bit assertion mechanism is the same as the interrupt generation mechanism described above for FREQIE.

Each FIFO is an 8-word FIFO. Each read or write operation from/to the FIFO targets one word FIFO location whatever the access size. Each FIFO word contains one audio slot. FIFO pointers are incremented by one word after each access to the SAI_xDR register.

Data must be right-aligned when written in the SAI_xDR register.

Data received are right-aligned in the SAI_xDR register.

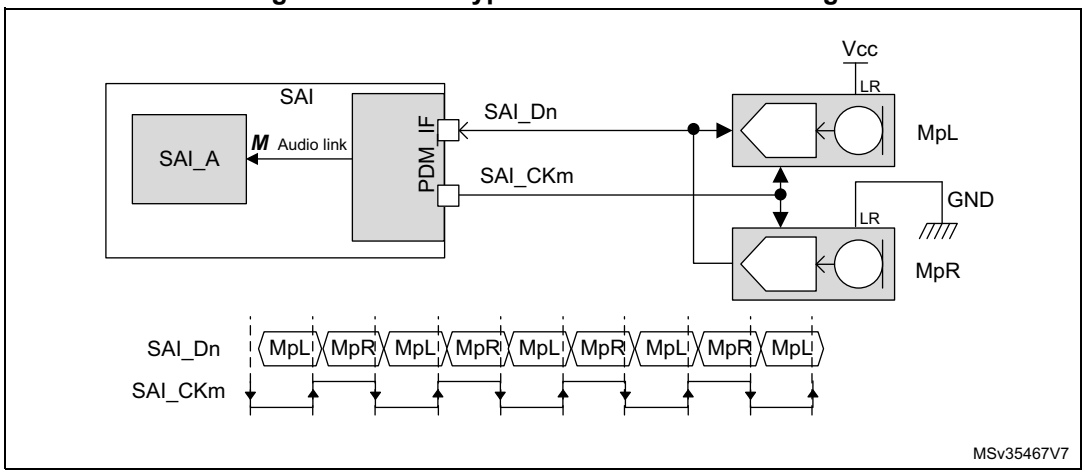
The FIFO pointers can be reinitialized when the SAI is disabled by configuring the FFLUSH bit in the SAI_xCR2 register. If FFLUSH is set when the SAI is enabled, the data present in the FIFO are lost automatically.

39.4.10 PDM interface

The PDM (pulse density modulation) interface is provided in order to support digital microphones. Up to 4 digital microphone pairs can be connected in parallel. Depending on product implementation, less microphones can be supported (refer to *Section 39.3: SAI implementation*).

Figure 363 shows a typical connection of a digital microphone pair via a PDM interface. Both microphones share the same bitstream clock and data line. Thanks to a configuration pin (LR), a microphone can provide valid data on SAI_CK[m] rising edge while the other provides valid data on SAI_CK[m] falling edge (m being the number of clock lines).

Figure 363. PDM typical connection and timing



1. **n** refers to the number of data lines and **p** to the number of microphone pairs.

The PDM function is intended to be used in conjunction with SAI_A subblock configured in TDM master mode. It cannot be used with SAI_B subblock. The PDM interface uses the timing signals provided by the serial audio interface of SAI_A and adapts them to generate a bitstream clock (SAI_CK[m]).

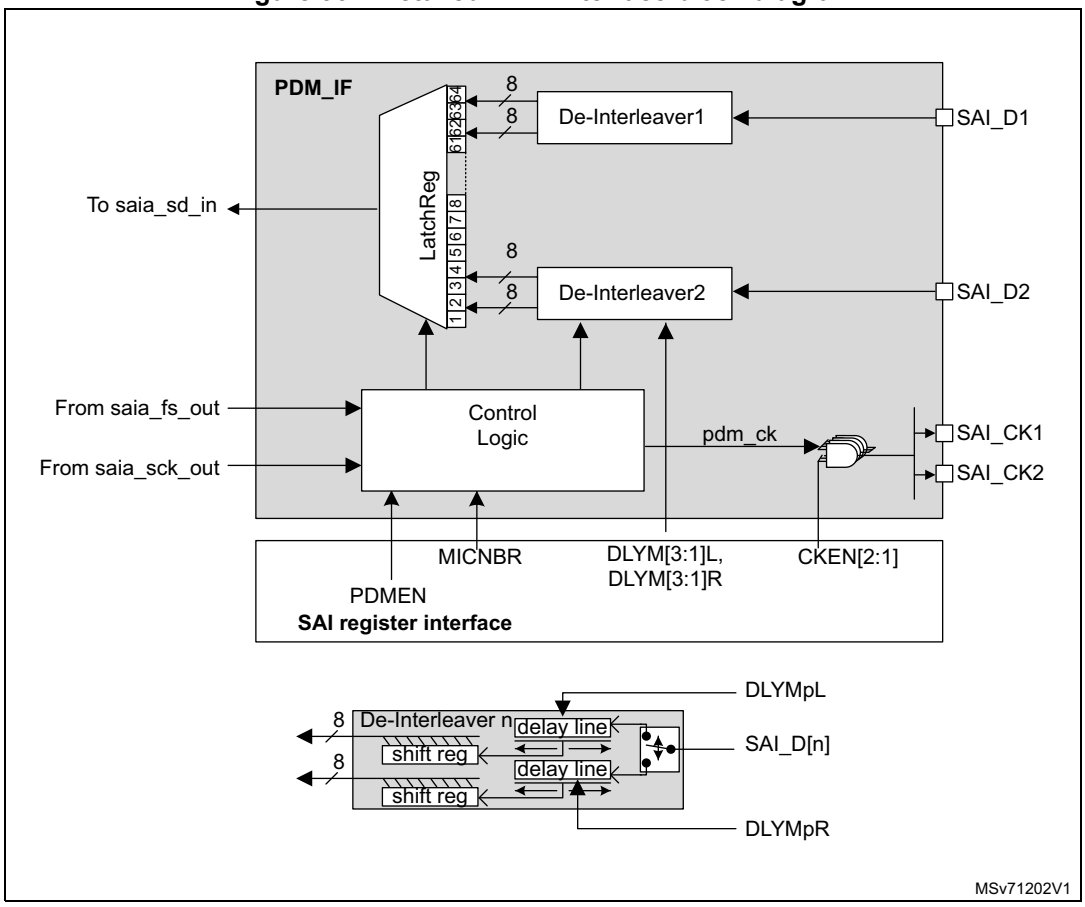
The processing performed into the PDM interface is the following:

1. The PDM interface builds the bitstream clock from the bit clock received from the serial audio interface of SAI_A.
2. The bitstream data received from the microphones (SAI_D[n]) are de-interleaved and go through a 7-bit delay line to fine-tune the delay of each microphone with the accuracy of the bitstream clock.
3. The shift registers translate each serial bitstream into bytes.
4. The last operation consists in shifting-out the resulting bytes to SAI_A through the data line of the serial audio interface.

Figure 364 below shows the block diagram of the PDM interface, with a detailed view of a de-interleaver.

Note: The PDM interface does not embed the decimation filter required to build-up the PCM audio samples from the bitstream. It is up to the application software to perform this operation.

Figure 364. Detailed PDM interface block diagram



1. **n** refers to the number of data lines and **p** to the number of microphone pairs.

The PDM interface can be enabled through the PDMEN bit in the SAI_PDMCR register. However, the PDM interface must be enabled prior to enabling the SAI_A block.

To reduce the memory footprint, the user can select the number of microphones the application needs. This can be done through the MICNBR[1:0] bits. It is possible to choose

between 2, 4, 6, or 8 microphones. For example, if the application is using 3 microphones, the user has to select 4.

Enabling the PDM interface

To enable the PDM interface, follow the sequence below:

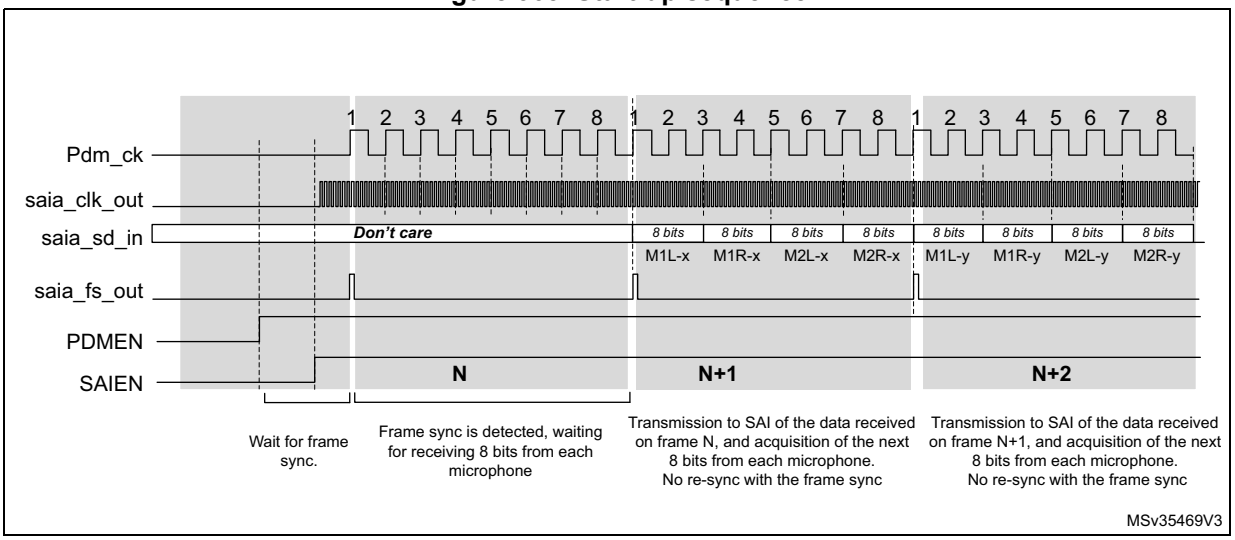
1. Configure SAI_A in TDM master mode (see *Table 377*).
2. Configure the PDM interface as follows:
   a) Define the number of digital microphones via MICNBR.
   b) Enable the bitstream clock needed in the application by setting the corresponding bits on CKEN to 1.
3. Enable the PDM interface, via the PDMEN bit.
4. Enable the SAI_A.

Note: Once the PDM interface and SAI_A are enabled, the first two frames received on SAI_ADR are invalid and must be dropped.

Startup sequence

Figure 365 shows the startup sequence: Once the PDM interface is enabled, it waits for the frame synchronization event prior to starting the acquisition of the microphone samples. After 8 SAI_CK clock periods, a data byte coming from each microphone is available, and transferred to the SAI, via the serial audio interface.

Figure 365. Start-up sequence



SAI_ADR data format

The arrangement of the data coming from the microphone into the SAI_ADR register depends on the following parameters:

- Number of microphones
- Slot width selected
- LSBFIRST bit

The slot width defines the number of significant bits into each word available into the SAI_ADR.

When a slot width of 32 bits is selected, each data available into the SAI_ADR contains 32 useful bits. This reduces the number of words stored into the memory. However, the counterpart is that the software has to perform some operations to de-interleave the data of each microphone.

On the other hand, when the slot width is set to 8 bits, each data available into the SAI_ADR contains 8 useful bits. This increases the number of words stored in the memory. However, it offers the advantage of avoiding extra processing since each word contains information from one microphone.

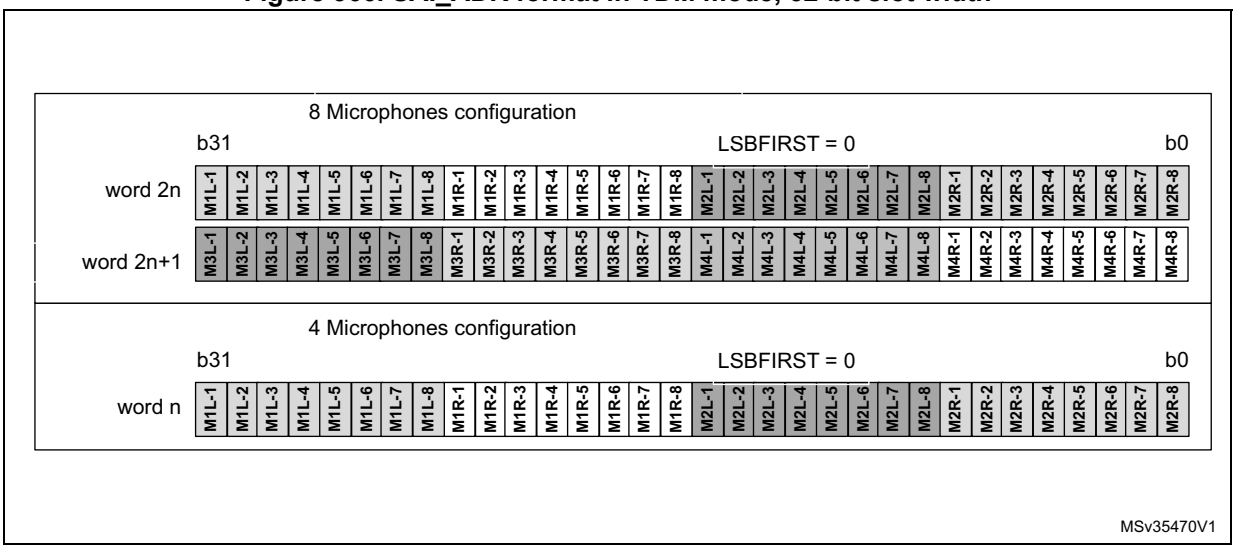
SAI_ADR data format example

- **32-bit slot width** (DS = 0b111 and SLOTSZ = 0). Refer to *Figure 366*.

  For an 8 microphone configuration, two consecutive words read from the SAI_ADR register contain a data byte from each microphone.

  For a 4 microphones configuration, each word read from the SAI_ADR register contains a data byte from each microphone.

Figure 366. SAI_ADR format in TDM mode, 32-bit slot width

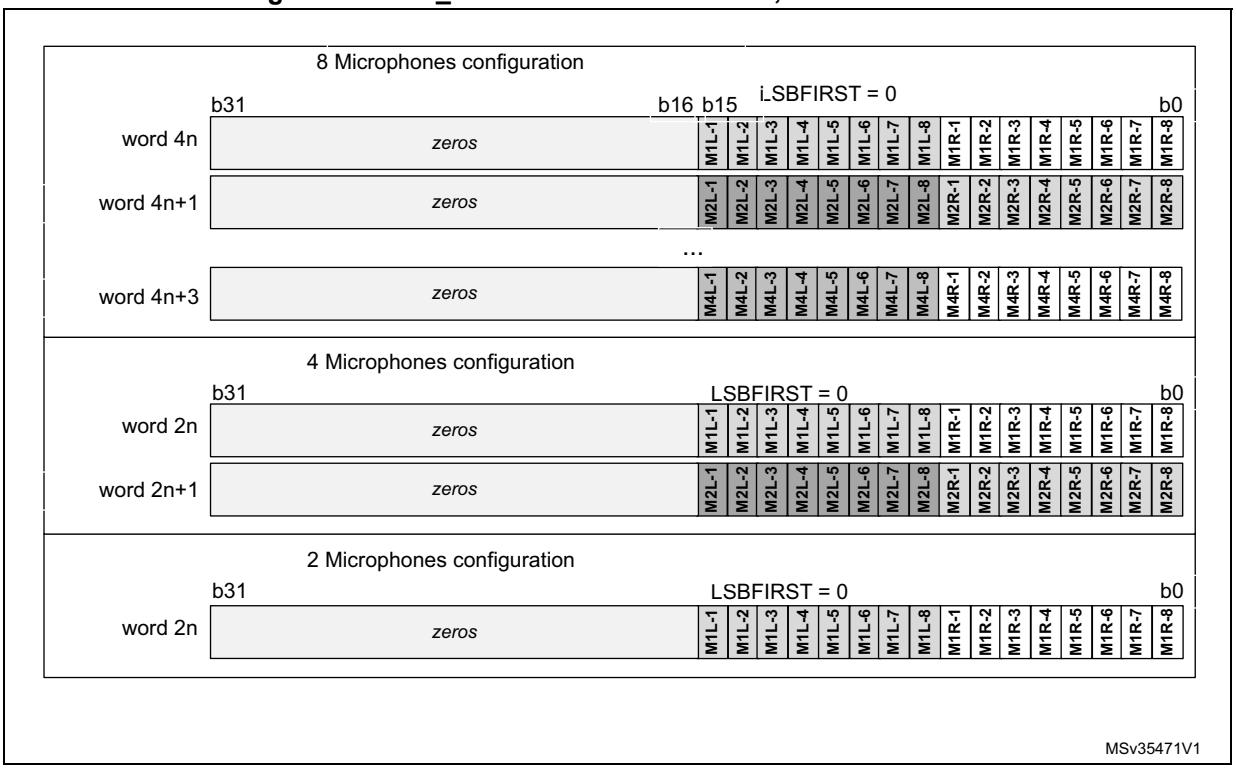


- **16-bit slot width** (DS = 0b100 and SLOTSZ = 0). Refer to *Figure 367*.

  For an 8-microphone configuration, four consecutive words read from the SAI_ADR register contain a data byte from each microphone. Note that the 16-bit data of SAI_ADR are right-aligned.

  For a 4- or 2-microphone configuration, the SAI behavior is similar to the 8-microphone configuration. Up to 2 words of 16 bits are required to acquire a byte from 4 microphones and a single word for 2 microphones.

Figure 367. SAI_ADR format in TDM mode, 16-bit slot width

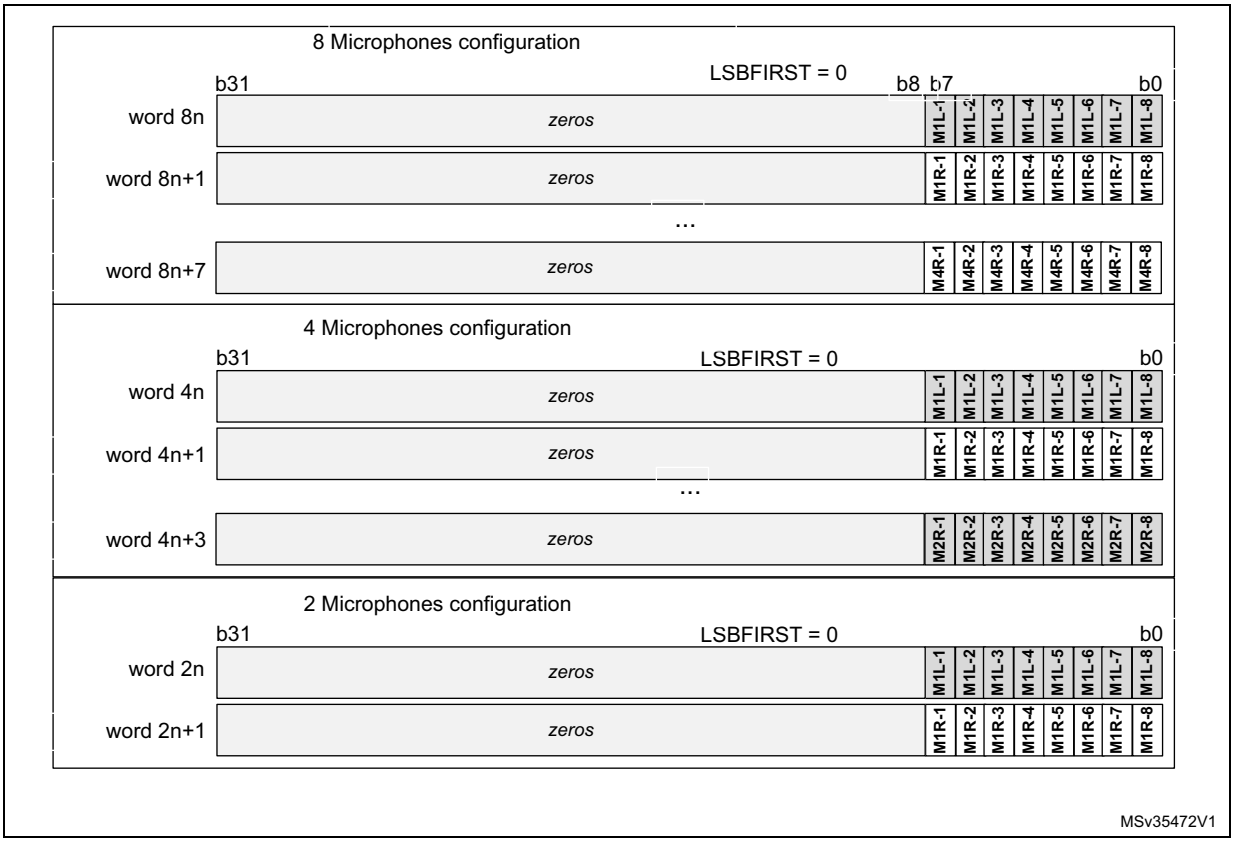


- **Using an 8-bit slot width** (DS = 0b010 and SLOTSZ = 0). Refer to *Figure 368*.

  For an 8-microphone configuration, eight consecutive words read from the SAI_ADR register contain a byte of data from each microphone. Note that the 8-bit data of SAI_ADR are right-aligned.

  For a 4- or 2-microphone configuration, the SAI behavior is similar to the 8-microphone configuration. Up to four words of eight bits are required to acquire a byte from four microphones and two words from two microphones.

Figure 368. SAI_ADR format in TDM mode, 8-bit slot width



TDM mode configuration for PDM interface

The SAI_A serial audio interface is internally connected to the PDM interface to get the microphone samples. The user application must configure the SAI_A serial audio interface as shown in *Table 377* to ensure a good connection with the PDM interface.

Table 377. SAI_A configuration for TDM mode

Bit Fields	Values	Comments
MODE	0b01	Mode must be MASTER receiver.
PRTCFG	0b00	Free protocol for TDM.
DS	X	To be adjusted according to the required data format, in accordance with the frame length and the number of slots (FRL and NBSLOT). See *Table 378*.
LSBFIRST	X	This parameter can be used according to the desired data format.
CKSTR	0	Signal transitions occur on the rising edge of the SCK_A bit clock. Signals are stable on the falling edge of the bit clock.
MONO	0	Stereo mode.
FRL	X	To be adjusted according to the number of microphones (MICNBR). See *Table 378*.
FSALL	0	Pulse width is one bit clock cycle.

Table 377. SAI_A configuration for TDM mode (continued)

Bit Fields	Values	Comments
FSDEF	0	FS signal is a start of frame.
FSPOL	1	FS is active high.
FSOFF	0	FS is asserted on the first bit of slot 0.
FBOFF	0	No offset on slot.
SLOTSZ	0	Slot size = data size.
NBSLOT	X	To be adjusted according to the required data format, in accordance with the slot size, and the frame length (FRL and DS). See *Table 378*.
SLOTEN	X	To be adjusted according to NBSLOT.
NODIV	1	No need to generate a master clock MCLK.
MCKDIV	X	Depends on the frequency provided to sai_a_ker_ck input. This parameter must be adjusted to generate the proper bitstream clock frequency. See *Table 378*.

Adjusting the bitstream clock rate

To program the serial audio interface properly, the user application must take into account the settings given in *Table 377*, and follow the sequence below:

1. Adjust the bit clock frequency (F_{SCK_A}) according to the required frequency for the PDM bitstream clock, using the following formula:

$$F_{SCK_A} = F_{PDM_CK} \times (MICNBR + 1) \times 2$$

   MICNBR can be 0, 1, 2 or 3 (0 = 2 microphones; see *Section 39.6.17*)

2. Set the frame length (FRL) using the following formula:

$$FRL = (16 \times (MICNBR + 1)) - 1$$

Table 378. TDM frame configuration examples(1)(2)

Microphone sampling rate	Nber of microphones	Wanted SAI_CKn frequency	bit clock (SCK_A) frequency	Frame sync. (FS_A) frequency	FRL	DS	NBSLOT	Comments
48 kHz	up to 8	3.072 MHz	24.576 MHz	384 kHz	63	0b111	1	2 slots of 32 bits per frame
		3.072 MHz	24.576 MHz	384 kHz	63	0b100	3	4 slots of 16 bits per frame
		3.072 MHz	24.576 MHz	384 kHz	63	0b010	7	8 slots of 8 bits per frame
	up to 6	3.072 MHz	18.432 MHz	384 kHz	47	0b110	1	2 slots of 24 bits per frame
		3.072 MHz	18.432 MHz	384 kHz	47	0b100	2	3 slots of 16 bits per frame
		3.072 MHz	18.432 MHz	384 kHz	47	0b010	5	6 slots of 8 bits per frame
	up to 4	3.072 MHz	12.288 MHz	384 kHz	31	0b111	0	1 slot of 32 bits per frame
		3.072 MHz	12.288 MHz	384 kHz	31	0b100	1	2 slots of 16 bits per frame
		3.072 MHz	12.288 MHz	384 kHz	31	0b010	3	4 slots of 8 bits per frame
	up to 2	3.072 MHz	6.144 MHz	384 kHz	15	0b100	0	1 slots of 16 bits per frame
		3.072 MHz	6.144 MHz	384 kHz	15	0b010	1	2 slots of 8 bits per frame
16 kHz	up to 8	1.024 MHz	8.192 MHz	128 kHz	63	0b111	1	2 slots of 32 bits per frame
		1.024 MHz	8.192 MHz	128 kHz	63	0b100	3	4 slots of 16 bits per frame
		1.024 MHz	8.192 MHz	128 kHz	63	0b010	7	8 slots of 8 bits per frame
	up to 6	1.024 MHz	6.144 MHz	128 kHz	47	0b110	1	2 slots of 24 bits per frame
		1.024 MHz	6.144 MHz	128 kHz	47	0b010	5	6 slots of 8 bits per frame
	up to 4	1.024 MHz	4.096 MHz	128 kHz	31	0b111	0	1 slot of 32 bits per frame
		1.024 MHz	4.096 MHz	128 kHz	31	0b100	1	2 slots of 16 bits per frame
		1.024 MHz	4.096 MHz	128 kHz	31	0b010	3	4 slots of 8 bits per frame
	up to 2	1.024 MHz	2.048 MHz	128 kHz	15	0b100	0	1 slot of 16 bits per frame
		1.024 MHz	2.048 MHz	128 kHz	15	0b010	1	2 slots of 8 bits per frame

1. Refer to *Table 377: SAI_A configuration for TDM mode* for additional information on TDM configuration. The sai_a_ker_ck clock frequency provided to the SAI must be a multiple of the SCK_A frequency, and MCKDIV must be programmed accordingly.
2. The above sai_a_ker_ck frequencies are given as examples only. Refer to section *Reset and clock controller (RCC) to check if they can be generated on the device*.
3. The table above gives allowed settings for a decimation ratio of 64.

Adjusting the delay lines

When the PDM interface is enabled, the application can adjust on-the-fly the delay cells of each microphone input via the SAI_PDMDLY register.

The new delay values become effective after two audio frames.

39.4.11 AC’97 link controller

The SAI is able to work as an AC’97 link controller. In this protocol:

- The slot number and the slot size are fixed.
- The frame synchronization signal is perfectly defined and has a fixed shape.

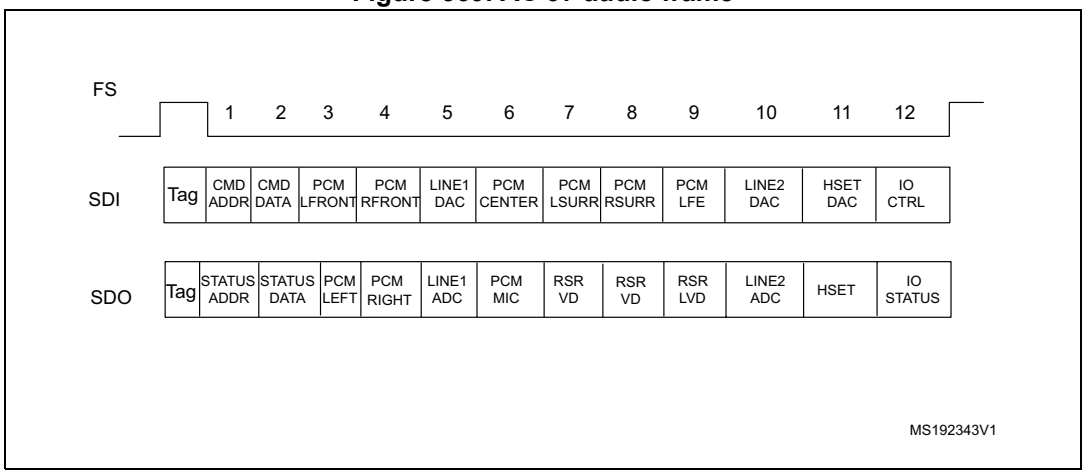
To select this protocol, set the PRTCFG[1:0] bits in the SAI_xCR1 register to 10. When AC’97 mode is selected, only data sizes of 16 or 20 bits can be used, otherwise the SAI behavior is not guaranteed.

- The NBSLOT[3:0] and SLOTSZ[1:0] bits are consequently ignored.
- The number of slots is fixed at 13 slots. The first one is 16 bits wide and all the others are 20 bits wide (data slots).
- The FBOFF[4:0] bits in the SAI_xSLOTR register are ignored.
- The SAI_xFRCR register is ignored.
- The MCLK is not used.

The FS signal from the block defined as asynchronous is configured automatically as an output, since the AC’97 controller link drives the FS signal whatever the master or slave configuration.

Figure 369 shows an AC’97 audio frame structure.

Figure 369. AC’97 audio frame



Note: In the AC’97 protocol, bit 2 of the tag is reserved (always 0), so bit 2 of the TAG is forced to 0 level whatever the value written in the SAI FIFO.

For more details about tag representation, refer to the AC’97 protocol standard.

One SAI can be used to target an AC’97 point-to-point communication.

In receiver mode, the SAI acting as an AC’97 link controller requires no FIFO request and so no data storage in the FIFO when the codec-ready bit in slot 0 is decoded low. If bit CNRDYIE is enabled in the SAI_xIM register, flag CNRDY is set in the SAI_xSR register and an interrupt is generated. This flag is dedicated to the AC’97 protocol.

Clock generator programming in AC'97 mode

In AC'97 mode, the frame length is fixed at 256 bits, and its frequency must be set to 48 kHz. The formulas given in *Section 39.4.8: SAI clock generator* must be used with FRL = 255, to generate the proper frame rate (F_{FS_x}).

39.4.12 SPDIF output

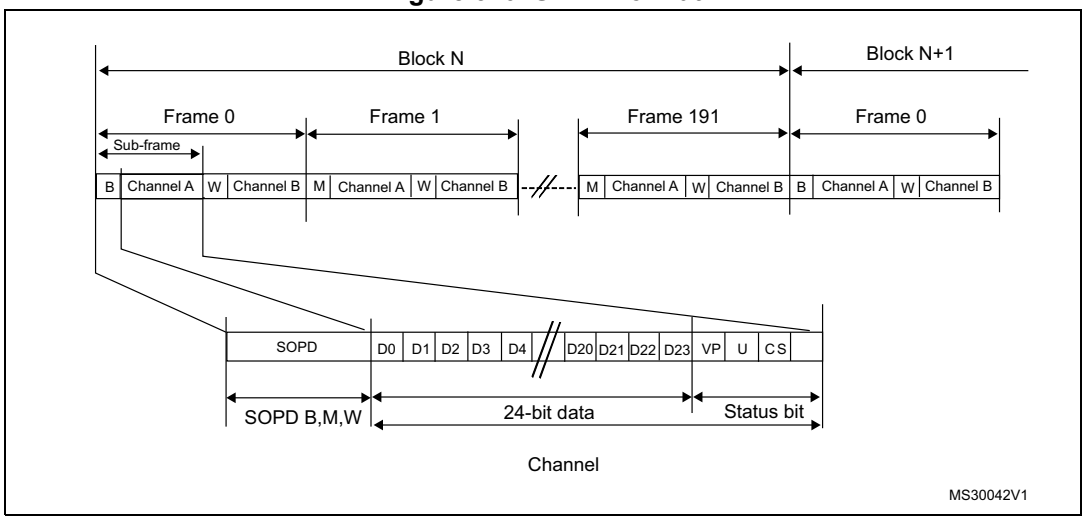
The SPDIF interface is available in transmitter mode only. It supports the audio IEC60958.

To select SPDIF mode, set the PRTCFG[1:0] bits to 01 in the SAI_xCR1 register.

For SPDIF protocol:

- Only the SD data line is enabled.
- The FS, SCK, and MCLK I/Os pins are left free.
- The MODE[1] bit is forced to 0 to select the master mode to enable the clock generator of the SAI and manage the data rate on the SD line.
- The data size is forced to 24 bits. The value set in the DS[2:0] bits in the SAI_xCR1 register is ignored.
- The clock generator must be configured to define the symbol-rate, knowing that the bit clock must be twice the symbol-rate. The data is coded in Manchester protocol.
- The SAI_xFRCR and SAI_xSLOTR registers are ignored. The SAI is configured internally to match the SPDIF protocol requirements as shown in *Figure 370*.

Figure 370. SPDIF format



An SPDIF block contains 192 frames. Each frame is composed of two 32-bit subframes, generally one for the left channel and one for the right channel. Each subframe is composed of a SOPD pattern (4-bit) to specify if the subframe is the start of a block (and so is identifying a channel A) or if it is identifying a channel A somewhere in the block, or if it is referring to channel B (see *Table 379*). The next 28 bits of channel information are composed of 24 data bits + 4 status bits.

Table 379. SOPD pattern

SOPD	Preamble coding		Description
	last bit is 0	last bit is 1	
B	11101000	00010111	Channel A data at the start of block
W	11100100	00011011	Channel B data somewhere in the block
M	11100010	00011101	Channel A data

The data stored in SAI_xDR has to be filled as follows:

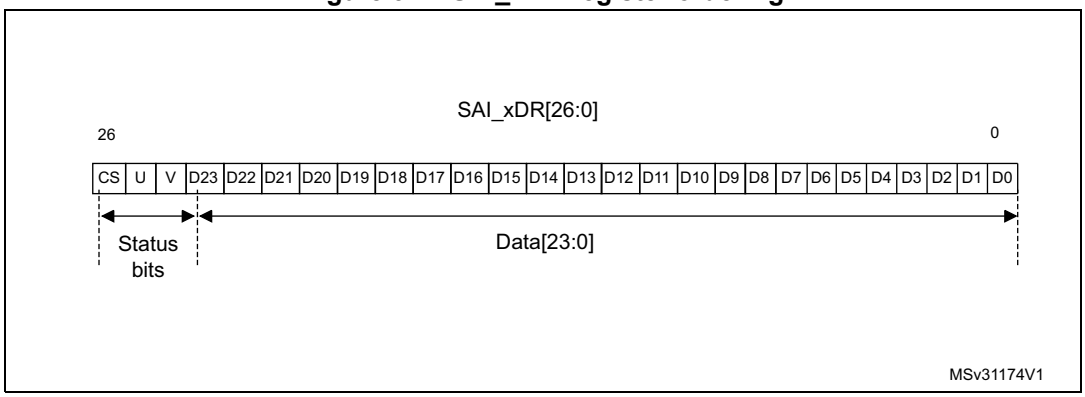
- SAI_xDR[26:24] contain the channel status, user, and validity bits.
- SAI_xDR[23:0] contain the 24-bit data for the considered channel.

If the data size is 20 bits, the data must be mapped on SAI_xDR[23:4].

If the data size is 16 bits, the data must be mapped on SAI_xDR[23:8].

SAI_xDR[23] always represents the MSB.

Figure 371. SAI_xDR register ordering



Note: The transfer is always performed with LSB first.

The SAI first sends the adequate preamble for each subframe in a block. The SAI_xDR is then sent on the SD line (Manchester coded). The SAI ends the subframe by transferring the parity bit calculated as described in *Table 380*.

Table 380. Parity bit calculation

SAI_xDR[26:0]	Parity bit P value transferred
odd number of 0	0
odd number of 1	1

The underrun is the only error flag available in the SAI_xSR register for SPDIF mode since the SAI can only operate in transmitter mode. As a result, the following sequence must be executed to recover from an underrun error detected via the underrun interrupt or the underrun status bit:

1. Disable the DMA stream (via the DMA peripheral) if the DMA is used.
2. Disable the SAI and check that the peripheral is physically disabled by polling the SAIEN bit in the SAI_xCR1 register.
3. Clear the COVRUNDR flag in the SAI_xCLRFR register.
4. Flush the FIFO by setting the FFLUSH bit in SAI_xCR2.

   The software needs to point to the address of the future data corresponding to the start of a new block (data for preamble B). If the DMA is used, the DMA source base address pointer must be updated accordingly.
5. Enable the DMA stream (DMA peripheral) again if the DMA is used to manage data transfers according to the new source base address.
6. Enable the SAI again by configuring the SAIEN bit in the SAI_xCR1 register.

Clock generator programming in SPDIF generator mode

For the SPDIF generator, the SAI provides a bit clock twice as fast as the symbol-rate. The table below shows examples of symbol rates with respect to the audio sampling rate.

Table 381. Audio sampling frequency versus symbol rates

Audio sampling frequencies (F_S)	Symbol-rate
44.1 kHz	2.8224 MHz
48 kHz	3.072 MHz
96 kHz	6.144 MHz
192 kHz	12.288 MHz

More generally, the relationship between the audio sampling frequency (F_S) and the bit clock rate (F_{SCK_X}) is given by the formula:

$$F_S = \frac{F_{SCK_x}}{128}$$

The bit clock rate is obtained as follows:

$$F_{SCK_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV}$$

Note: The above formulas are valid only if NODIV is set to 1 in the SAI_ACR1 register.

39.4.13 Specific features

The SAI interface embeds specific features that can be useful depending on the audio protocol selected. These functions are accessible through specific bits of the SAI_xCR2 register.

Mute mode

The mute mode can be used when the audio subblock is a transmitter or a receiver.

Audio subblock in transmission mode

In transmitter mode, the mute mode can be selected at any time. The mute mode is active for entire audio frames. The MUTE bit in the SAI_xCR2 register enables the mute mode when it is configured during an ongoing frame.

The mute mode bit is strobed only at the end of the frame. If it is set at this time, the mute mode is active at the beginning of the new audio frame and for a complete frame, until the next end of frame. The bit is then strobed to determine if the next frame is still a mute frame.

If the number of slots set through the NBSLOT[3:0] bits in the SAI_xSLOTR register is lower than or equal to 2, it is possible to specify if the value sent in mute mode is 0 or if it is the last value of each slot. The selection is done via the MUTEVAL bit in the SAI_xCR2 register.

If the number of slots set in the NBSLOT[3:0] bits in the SAI_xSLOTR register is greater than 2, the MUTEVAL bit in the SAI_xCR2 register is meaningless as 0 values are sent on each bit on each slot.

The FIFO pointers are still incremented in mute mode. This means that data present in the FIFO and for which the mute mode is requested are discarded.

Audio subblock in reception mode

In reception mode, it is possible to detect a mute mode sent from the external transmitter when all the declared and valid slots of the audio frame receive 0 for a given consecutive number of audio frames (MUTECNT[5:0] bits in the SAI_xCR2 register).

When the number of MUTE frames is detected, the MUTEDET flag in the SAI_xSR register is set and an interrupt can be generated if the MUTEDETIE bit is set in SAI_xCR2.

The mute frame counter is cleared when the audio subblock is disabled or when a valid slot receives at least one data in an audio frame. The interrupt is generated just once, when the counter reaches the value specified in the MUTECNT[5:0] bits. The interrupt event is then reinitialized when the counter is cleared.

Note: The mute mode is not available for SPDIF audio blocks.

Mono/stereo mode

In transmitter mode, the mono mode can be addressed without any data preprocessing in memory, assuming the number of slots is equal to 2 (NBSLOT[3:0] = 0001 in SAI_xSLOTR). In this case, the access time to and from the FIFO is reduced by 2 since the data for slot 0 is duplicated into data slot 1.

To enable the mono mode:

1. Set the MONO bit to 1 in the SAI_xCR1 register.
2. Set NBSLOT to 1 and SLOTEN to 3 in SAI_xSLOTR.

In reception mode, the MONO bit can be set and is meaningful only if the number of slots is equal to 2, like in transmitter mode. When it is set, only slot 0 data are stored in the FIFO. The data belonging to slot 1 are discarded since, in this case, it is supposed to be the same as the previous slot. If the data flow in reception mode is a real stereo audio flow with a distinct and different left and right data, the MONO bit is meaningless. The conversion from the output stereo file to the equivalent mono file is done by software.

Companding mode

Telecommunication applications can require processing the data to be transmitted or received using a data companding algorithm.

Depending on the COMP[1:0] bits in the SAI_xCR2 register (used only when free protocol mode is selected), the application software can choose to process or not the data before sending it on the SD serial output line (compression) or to expand the data after the reception on the SD serial input line (expansion), as illustrated in *Figure 372*. The two companding modes supported are the µ-Law and the A-Law logs, which are a part of the CCITT G.711 recommendation.

The companding standard used in the United States and Japan is the µ-Law. It supports 14 bits of dynamic range (COMP[1:0] = 10 in the SAI_xCR2 register).

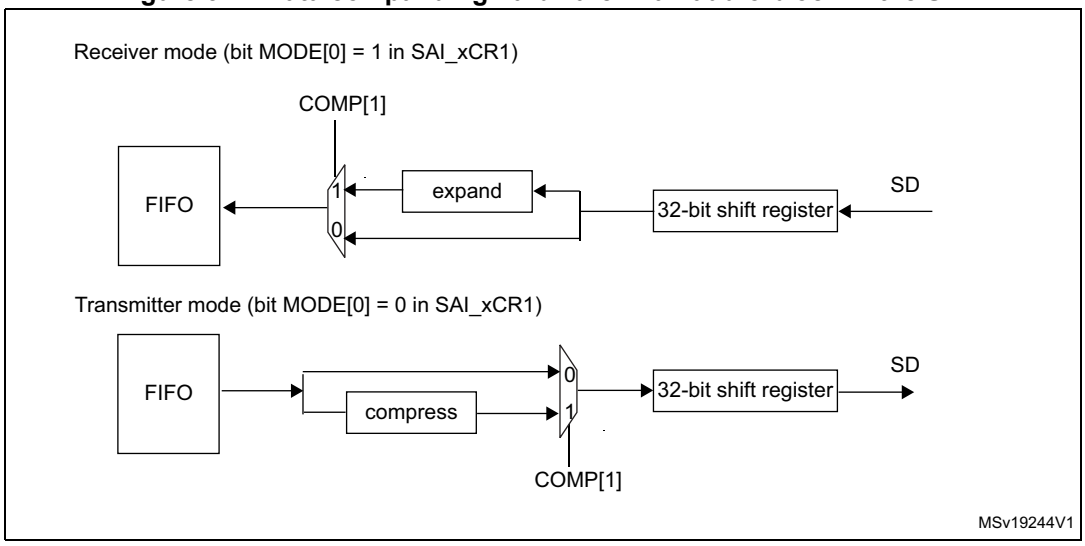
The European companding standard is A-Law and supports 13 bits of dynamic range (COMP[1:0] = 11 in the SAI_xCR2 register).

Both µ-Law and A-Law companding standard can be computed based on 1's complement or 2's complement representation, depending on the CPL bit setting in the SAI_xCR2 register.

In µ-Law and A-Law standards, data are coded as 8 bits with MSB alignment. Companded data are always 8 bits wide. For this reason, the DS[2:0] bits in the SAI_xCR1 register are forced to 010 when the SAI audio block is enabled (the SAIEN bit = 1 in the SAI_xCR1 register) and when one of these two companding modes is selected through the COMP[1:0] bits.

If no companding processing is required, the COMP[1:0] bits must be kept clear.

Figure 372. Data companding hardware in an audio block in the SAI



1. Not applicable when AC'97 or SPDIF are selected.

Expansion and compression mode are automatically selected through SAI_xCR2:

- If the SAI audio block is configured to be a transmitter, and if the COMP[1] bit is set in the SAI_xCR2 register, the compression mode is applied.
- If the SAI audio block is declared as a receiver, the expansion algorithm is applied.

Output data line management on an inactive slot

In transmitter mode, it is possible to choose the behavior of the SD line output when an inactive slot is sent on the data line (via the TRIS bit).

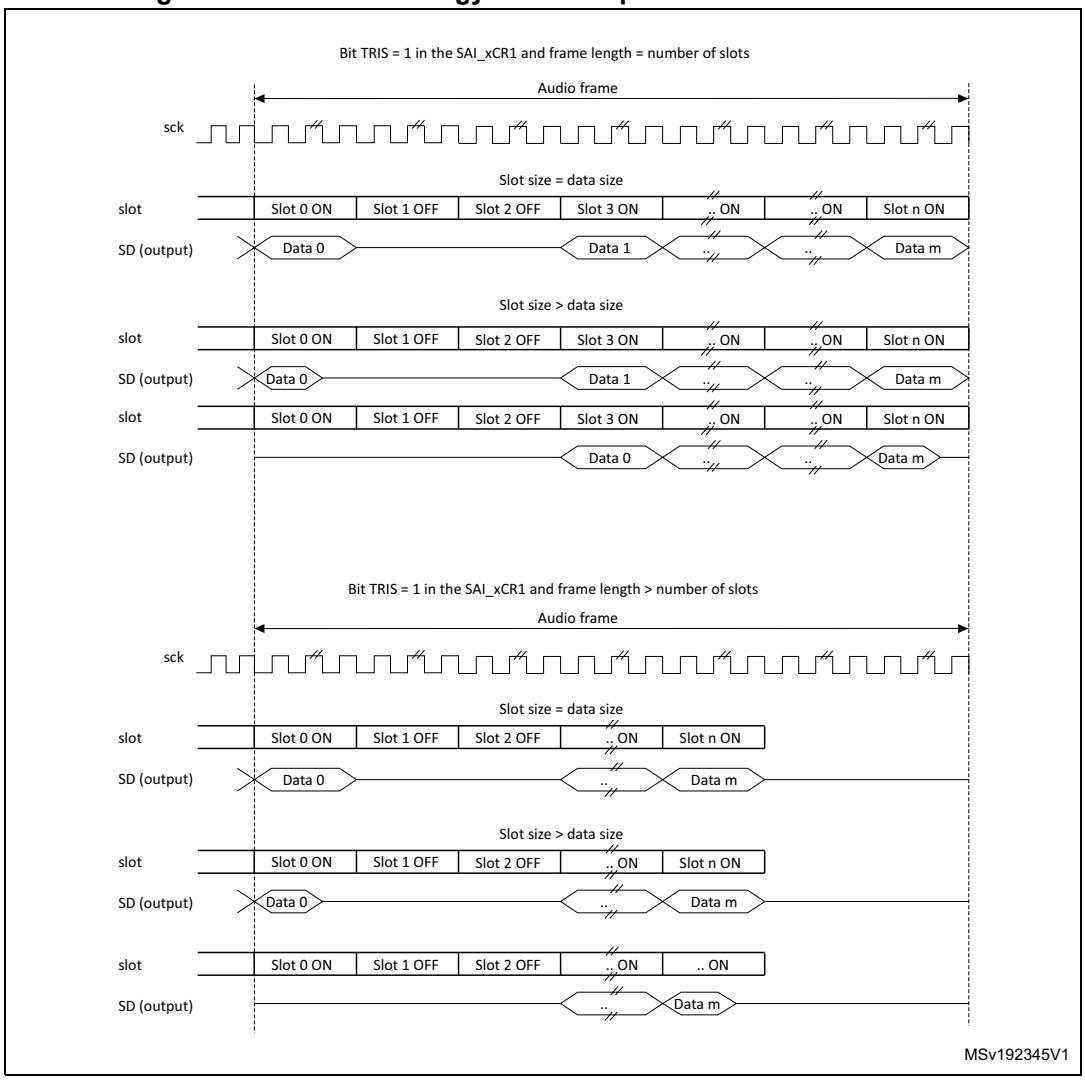
- Either the SAI forces 0 on the SD output line when an inactive slot is transmitted, or
- The line is released in high-Z state at the end of the last bit of data transferred, to release the line for other transmitters connected to this node.

It is important to note that the two transmitters cannot attempt to drive the same SD output pin simultaneously, which may result in a short circuit. To ensure a gap between transmissions, if the data is lower than 32-bit, the data can be extended to 32-bit by setting the bit SLOTSZ[1:0] = 10 in the SAI_xSLOTR register. The SD output pin is then tri-stated at the end of the LSB of the active slot (during the padding to 0 phase to extend the data to 32-bit) if the following slot is declared inactive.

In addition, if the number of slots multiplied by the slot size is lower than the frame length, the SD output line is tri-stated when the padding to 0 is done to complete the audio frame.

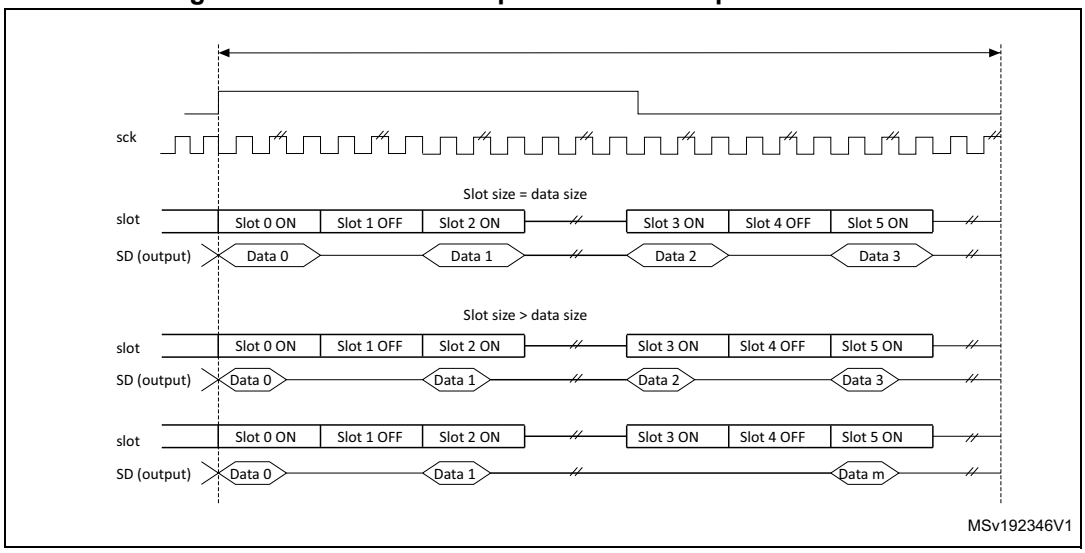
Figure 373 illustrates these behaviors.

Figure 373. Tristate strategy on SD output line on an inactive slot



When the selected audio protocol uses the FS signal as a start of frame and a channel side identification (bit FSDEF = 1 in the SAI_xFRCR register), the tristate mode is managed according to *Figure 374* (where the bit TRIS in the SAI_xCR1 register = 1, and FSDEF=1, and half frame length is higher than number of slots/2, and NBSLOT=6).

Figure 374. Tristate on output data line in a protocol like I2S



If the TRIS bit in the SAI_xCR2 register is cleared, all the high impedance states on the SD output line in *Figure 373* and *Figure 374* are replaced by a drive with a value of 0.

39.4.14 Error flags

The SAI implements the following error flags:

- FIFO overrun/underrun.
- Anticipated frame synchronization detection.
- Late frame synchronization detection.
- Codec not ready (AC'97 exclusively).
- Wrong clock configuration in master mode.

FIFO overrun/underrun (OVRUDR)

The FIFO overrun/underrun bit is called OVRUDR in the SAI_xSR register.

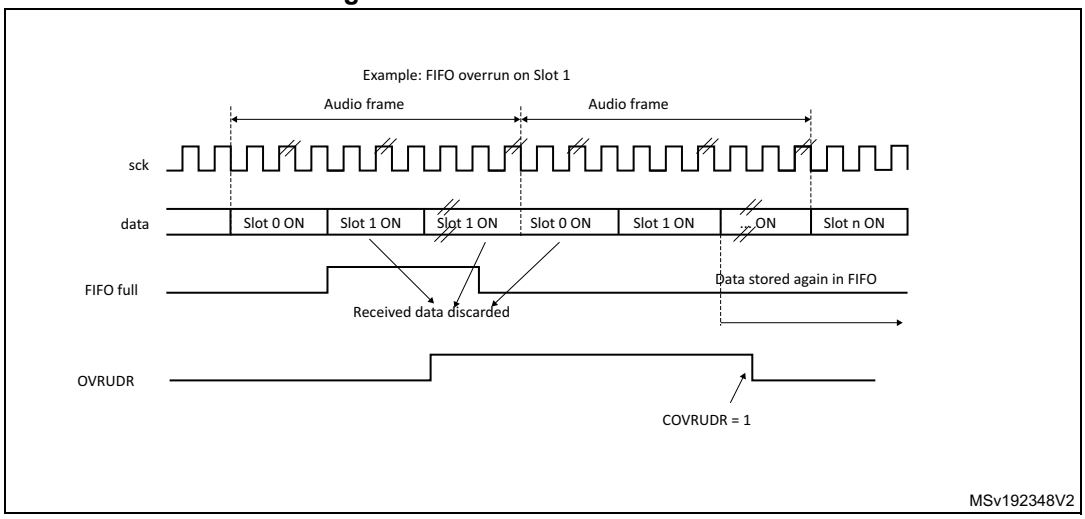
The overrun or underrun errors share the same bit since an audio block can be either receiver or transmitter and each audio block in a given SAI has its own SAI_xSR register.

Overrun

When the audio block is configured as receiver, an overrun condition may appear if data are received in an audio frame when the FIFO is full and not able to store the received data. In this case, the received data are lost, the OVRUDR flag in the SAI_xSR register is set, and an interrupt is generated if the OVRUDRIE bit is set in the SAI_xIM register. The slot number, from which the overrun occurs, is stored internally. No more data are stored into the FIFO until it becomes free to store new data. When the FIFO has at least one data free, the SAI audio block receiver stores new data (from a new audio frame) from the slot number that was stored internally when the overrun condition was detected. This avoids data slot dealignment in the destination memory (refer to *Figure 375*).

The OVRUDR flag is cleared when the COVRUDR bit is set in the SAI_xCLRFR register.

Figure 375. Overrun detection error



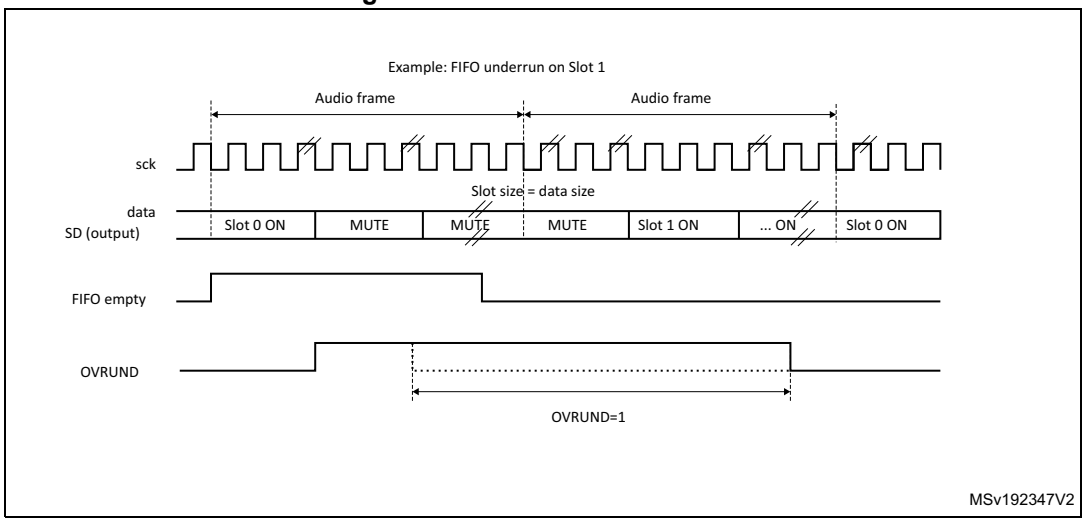
Underrun

An underrun may occur when the audio block in the SAI is a transmitter and the FIFO is empty when data need to be transmitted. If an underrun is detected, the slot number for which the event occurs is stored and the MUTE value (00) is sent until the FIFO is ready to transmit the data corresponding to the slot for which the underrun was detected (refer to *Figure 376*). This avoids desynchronization between the memory pointer and the slot in the audio frame.

The underrun event sets the OVRUDR flag in the SAI_xSR register and an interrupt is generated if the OVRUDRIE bit is set in the SAI_xIM register. To clear this flag, set the COVRUDR bit in the SAI_xCLRFR register.

The underrun event can occur when the audio subblock is configured as master or slave.

Figure 376. FIFO underrun event



Anticipated frame synchronization detection (AFSDET)

The AFSDET flag is used only in slave mode. It is never asserted in master mode. It indicates that a frame synchronization (FS) has been detected earlier than expected since the frame length, the frame polarity, and the frame offset are defined and known.

Anticipated frame detection sets the AFSDET flag in the SAI_xSR register.

This detection has no effect on the current audio frame, which is not sensitive to the anticipated FS. This means that “parasitic” events on signal FS are flagged without any perturbation of the current audio frame.

An interrupt is generated if the AFSDETIE bit is set in the SAI_xIM register. To clear the AFSDET flag, the CAFSDET bit must be set in the SAI_xCLRFR register.

To resynchronize with the master after an anticipated frame detection error, four steps are required:

1. Disable the SAI block by resetting the SAIEN bit in the SAI_xCR1 register. To make sure that the SAI is disabled, read back the SAIEN bit and check it is set to 0.
2. Flush the FIFO via the FFLUS bit in the SAI_xCR2 register.
3. Enable the SAI peripheral again (SAIEN bit set to 1).
4. The SAI block waits for the assertion on FS to restart the synchronization with master.

Note: The AFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave.It has no meaning in SPDIF mode since the FS signal is not used.

Late frame synchronization detection

The LFSDET flag in the SAI_xSR register can be set only when the SAI audio block operates as a slave. The frame length, the frame polarity, and the frame-offset configuration are known in register SAI_xFRCR.

If the external master does not send the FS signal at the expected time, thus generating the signal too late, the LFSDET flag is set and an interrupt is generated if the LFSDETIE bit is set in the SAI_xIM register.

The LFSDET flag is cleared when the CLFSDET bit is set in the SAI_xCLRFR register.

The late frame synchronization detection flag is set when the corresponding error is detected. The SAI needs to be resynchronized with the master (see sequence described in *Anticipated frame synchronization detection (AFSDET)*).

In a noisy environment, glitches on the SCK clock may be wrongly detected by the audio block state machine and shift the SAI data at a wrong frame position. This event can be detected by the SAI and reported as a late frame synchronization detection error.

There is no corruption if the external master is not managing the audio data frame transfer in continuous mode, which must not be the case in most applications. In this case, the LFSDET flag is set.

Note: The LFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave.It has no meaning in SPDIF mode since the signal FS is not used by the protocol.

Codec not ready (CNRDY AC'97)

The CNRDY flag in the SAI_xSR register is relevant only if the SAI audio block is configured to operate in AC'97 mode (PRTCFG[1:0] = 10 in the SAI_xCR1 register). If the CNRDYIE bit is set in the SAI_xIM register, an interrupt is generated when the CNRDY flag is set.

CNRDY is asserted when the codec is not ready to communicate during the reception of the TAG 0 (slot 0) of the AC'97 audio frame. In this case, no data are automatically stored into the FIFO since the codec is not ready, until the TAG 0 indicates that the codec is ready. All the active slots defined in the SAI_xSLOTR register are captured when the codec is ready.

To clear the CNRDY flag, the CCNRDY bit must be set in the SAI_xCLRFR register.

Wrong clock configuration in master mode (with NODIV = 0)

When the audio block operates as a master (MODE[1] = 0) and the NODIV bit is equal to 0, the WCKCFG flag is set as soon as the SAI is enabled if the following conditions are met:

- (FRL+1) is not a power of 2, and
- (FRL+1) is not between 8 and 256.

The MODE, NODIV, and SAIEN bits belong to the SAI_xCR1 register and FRL to the SAI_xFRCR register.

If the WCKCFGIE bit is set, an interrupt is generated when the WCKCFG flag is set in the SAI_xSR register. To clear this flag, set the CWCKCFG bit in the SAI_xCLRFR register.

When the WCKCFG bit is set, the audio block is automatically disabled, thus performing a hardware clear of the SAIEN bit.

39.4.15 Disabling the SAI

The SAI audio block can be disabled at any moment by clearing the SAIEN bit in the SAI_xCR1 register. All the already started frames are automatically completed before the SAI stops working. The SAIEN bit remains high until the SAI is completely switched off at the end of the current audio frame transfer.

If an audio block in the SAI operates synchronously with the other one, the one that is the master must be disabled first.

39.4.16 SAI DMA interface

To free the CPU and to optimize bus bandwidth, each SAI audio block has an independent DMA interface to read/write from/to the SAI_xDR register (to access the internal FIFO). There is one DMA channel per audio subblock supporting the basic DMA request/acknowledge protocol.

To configure the audio subblock for DMA transfer, set the DMAEN bit in the SAI_xCR1 register. The DMA request is managed directly by the FIFO controller depending on the FIFO threshold level (for more details refer to *Section 39.4.9: Internal FIFOs*). The DMA transfer direction is linked to the SAI audio subblock configuration:

- If the audio block operates as a transmitter, the audio block FIFO controller outputs a DMA request to load the FIFO with data written in the SAI_xDR register.
- If the audio block operates as a receiver, the DMA request is related to read operations from the SAI_xDR register.

Follow the sequence below to configure the SAI interface in DMA mode:

1. Configure the SAI and FIFO threshold levels to specify when the DMA request is launched.
2. Configure the SAI DMA channel.
3. Enable the DMA.
4. Enable the SAI interface.

39.5 SAI interrupts

The SAI supports 7 interrupt sources, as shown in *Table 382*.

Table 382. SAI interrupt sources

Interrupt acronym	Interrupt source	Interrupt group	Audio block mode	Interrupt enable	Interrupt clear
SAI	FREQ	FREQ	Master or slave Receiver or transmitter	FREQIE in SAI_xIM register	Depends on: – FIFO threshold setting (FLVL bits in SAI_xCR2) – Communication direction (transmitter or receiver) For more details refer to *Section 39.4.9: Internal FIFOs*
	OVRUDR	ERROR	Master or slave Receiver or transmitter	OVRUDRIE in SAI_xIM register	COVRUDR = 1 in SAI_xCLRFR register
	AFSDET	ERROR	Slave (not used in AC'97 mode and SPDIF mode)	AFSDETIE in SAI_xIM register	CAFSDET = 1 in SAI_xCLRFR register
	LFSDET	ERROR	Slave (not used in AC'97 mode and SPDIF mode)	LFSDETIE in SAI_xIM register	CLFSDET = 1 in SAI_xCLRFR register
	CNRDY	ERROR	Slave (only in AC'97 mode)	CNRDYIE in SAI_xIM register	CCNRDY = 1 in SAI_xCLRFR register
	MUTEDET	MUTE	Master or slave Receiver mode only	MUTEDETIE in SAI_xIM register	CMUTEDET = 1 in SAI_xCLRFR register
	WCKCFG	ERROR	Master with NODIV = 0 in SAI_xCR1 register	WCKCFGIE in SAI_xIM register	CWCKCFG = 1 in SAI_xCLRFR register

Follow the sequence below to enable an interrupt:

1. Disable SAI interrupt.
2. Configure SAI.
3. Configure SAI interrupt source.
4. Enable SAI.

39.6 SAI registers

The peripheral registers have to be accessed by words (32 bits).

39.6.1 SAI configuration register 1 (SAI_ACR1)

Address offset: 0x04

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN
				rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MCKEN:** Master clock generation enable
0: The master clock is not generated
1: The master clock is generated independently of SAIEN bit

Bit 26 **OSR:** Oversampling ratio for master clock
This bit is meaningful only when NODIV bit is set to 0.
0: Master clock frequency = F_{FS} x 256
1: Master clock frequency = F_{FS} x 512

Bits 25:20 **MCKDIV[5:0]**: Master clock divider
These bits are set and cleared by software.
000000: Divides by 1 the kernel clock input (sai_x_ker_ck).
Otherwise, The master clock frequency is calculated according to the formula given in *Section 39.4.8: SAI clock generator*.
These bits have no meaning when the audio block is slave.
They have to be configured when the audio block is disabled.

Bit 19 **NODIV:** No divider
This bit is set and cleared by software.
0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512
1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable
This bit is set and cleared by software.
0: DMA disabled
1: DMA enabled

Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.

Bit 16 **SAIEN**: Audio block enable

This bit is set by software.

To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.

This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.

0: SAI audio block disabled

1: SAI audio block enabled.

Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRIV**: Output drive

This bit is set and cleared by software.

0: Audio block output driven when SAIEN is set

1: Audio block output driven immediately after the setting of this bit.

Note: This bit has to be set before enabling the audio block and after the audio block configuration.

Bit 12 **MONO**: Mono mode

This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to *Section : Mono/stereo mode* for more details.

0: Stereo mode

1: Mono mode.

Bits 11:10 **SYNCEN[1:0]:** Synchronization enable

These bits are set and cleared by software. They must be configured when the audio subblock is disabled.

00: audio subblock in asynchronous mode.

01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode

10: Reserved.

11: Reserved

Note: The audio subblock must be configured as asynchronous when SPDIF mode is enabled.

Bit 9 **CKSTR:** Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.

1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST:** Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC'97 audio protocol since AC'97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

0: Data are transferred with MSB first

1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]:** Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

000: Reserved

001: Reserved

010: 8 bits

011: 10 bits

100: 16 bits

101: 20 bits

110: 24 bits

111: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]:** Protocol configuration

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

00: Free protocol. Free protocol enables to use the powerful configuration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.

01: SPDIF protocol

10: AC'97 protocol

11: Reserved

Bits 1:0 **MODE[1:0]:** SAIx audio block mode

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

00: Master transmitter

01: Master receiver

10: Slave transmitter

11: Slave receiver

Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00).

39.6.2 SAI configuration register 2 (SAI_ACR2)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[1:0]		CPL	MUTECNT[5:0]						MUTE VAL	MUTE	TRIS	F FLUSH	FTH[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **COMP[1:0]**: Companding mode.

These bits are set and cleared by software. The µ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on *CPL bit*.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to *Section : Companding mode* for more details.

00: No companding algorithm

01: Reserved.

10: µ-Law algorithm

11: A-Law algorithm

Note: Companding mode is applicable only when Free protocol mode is selected.

Bit 13 **CPL**: Complement bit.

This bit is set and cleared by software.

It defines the type of complement to be used for companding mode

0: 1's complement representation.

1: 2's complement representation.

Note: This bit has effect only when the companding mode is µ-Law algorithm or A-Law algorithm.

Bits 12:7 **MUTECNT[5:0]**: Mute counter.

These bits are set and cleared by software. They are used only in reception mode.

The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.

Refer to *Section : Mute mode* for more details.

Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.

If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.

if the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to *Section : Mute mode* for more details.

0: Bit value 0 is sent during the mute mode.

1: Last values are sent during the mute mode.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 5 **MUTE:** Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to *Section : Mute mode* for more details.

0: No mute mode.

1: Mute mode enabled.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It must be configured when SAI is disabled.

Refer to *Section : Output data line management on an inactive slot* for more details.

0: SD output line is still driven by the SAI when a slot is inactive.

1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH:** FIFO flush.

This bit is set by software. It is always read as 0. This bit must be configured when the SAI is disabled.

0: No FIFO flush.

1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled

Bits 2:0 **FTH[2:0]:** FIFO threshold.

This bit is set and cleared by software.

000: FIFO empty

001: ¼ FIFO

010: ½ FIFO

011: ¾ FIFO

100: FIFO full

101: Reserved

110: Reserved

111: Reserved

39.6.3 SAI frame configuration register (SAI_AFRCR)

Address offset: 0x0C

Reset value: 0x0000 0007

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF
													rw	rw	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FSALL[6:0]							FRL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **FSOFF**: Frame synchronization offset.

This bit is set and cleared by software. It is meaningless and is not used in AC'97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.

0: FS is asserted on the first bit of the slot 0.

1: FS is asserted one bit before the first bit of the slot 0.

Bit 17 **FSPOL**: Frame synchronization polarity.
This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC'97 or SPDIF audio block configuration.
This bit must be configured when the audio block is disabled.
0: FS is active low (falling edge)
1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.
This bit is set and cleared by software.
0: FS signal is a start frame signal
1: FS signal is a start of frame signal + channel side identification
When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots are dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).
This bit is meaningless and is not used in AC'97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.
These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame
These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.
They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.
These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.
The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).
In master mode, if the master clock (available on MCLK_x pin) is used, the frame length must be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.
These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration. They must be configured when the audio block is disabled.

39.6.4 SAI slot register (SAI_ASLOTR)

Address offset: 0x10

Reset value: 0x0000 0000

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOTEN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
				rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:16 **SLOTEN[15:0]**: Slot enable.
These bits are set and cleared by software.
Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).
0: Inactive slot.
1: Active slot.
The slot must be enabled when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.
These bits are set and cleared by software.
The value set in this bitfield represents the number of slots in the audio frame (including the number of inactive slots). The maximum number of slots is 16.
The number of slots must be even if FSDEF bit in the SAI_xFRCR register is set.
The number of slots must be configured when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
0000: Number of slots is 1.
0001: Number of slots is 2.
...
1111: Number of slots is 16.

Bits 7:6 **SLOTSZ[1:0]**: Slot size
This bits is set and cleared by software.
The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.
Refer to *Output data line management on an inactive slot* for information on how to drive SD line.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
01: 16-bit
10: 32-bit
11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset
These bits are set and cleared by software.
The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

39.6.5 SAI interrupt mask register (SAI_AIM)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.
This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.
This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC’97).
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC’97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.
This bit has a meaning only if the AC’97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode,

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.
It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.
Note: This bit is used only in Free protocol mode *and is meaningless in other modes.*

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

39.6.6 SAI status register (SAI_ASR)

Address offset: 0x18

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]		
													r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
									r	r	r	r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.
This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).
000: FIFO empty (transmitter and receiver modes)
001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
101: FIFO full (transmitter and receiver modes)
Others: Reserved

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **LFSDET**: Late frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is not present at the right time.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC’97 or SPDIF mode.
It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register

Bit 5 **AFSDET**: Anticipated frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is detected earlier than expected.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97 or SPDIF mode.
It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.
This bit is read only.
0: External AC'97 Codec is ready
1: External AC'97 Codec is not ready
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.
This bit is read only.
0: No FIFO request.
1: FIFO request to read or to write the SAI_xDR.
The request depends on the audio block configuration:
– If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
– If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.
This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.
This bit is read only.
0: Clock configuration is correct
1: Clock configuration does not respect the rule concerning the frame length specification defined in *Section 39.4.6: Frame synchronization* (configuration of FRL[7:0] bit in the SAI_xFRCR register)
This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.
It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.
This bit is read only.
0: No MUTE detection on the SD input line
1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame
This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).
It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets bit CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.
This bit is read only.
0: No overrun/underrun error.
1: Overrun/underrun error detection.
The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.
It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.
This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

39.6.7 SAI clear flag register (SAI_ACLRFR)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
									w	w	w		w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC’97 or SPDIF mode
Reading this bit always returns the value 0.

Bit 5 **CAFSDET**: Clear anticipated frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC’97 or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.
This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

39.6.8 SAI data register (SAI_ADR)

Address offset: 0x20

Reset value: 0x0000 0000

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

Bits 31:0 **DATA[31:0]**: Data
A write to this register loads the FIFO provided the FIFO is not full.
A read from this register empties the FIFO if the FIFO is not empty.

39.6.9 SAI configuration register 1 (SAI_BCR1)

Address offset: 0x24

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN
				rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MCKEN:** Master clock generation enable
0: The master clock is not generated
1: The master clock is generated independently of SAIEN bit

Bit 26 **OSR:** Oversampling ratio for master clock
This bit is meaningful only when NODIV bit is set to 0.
0: Master clock frequency = F_{FS} x 256
1: Master clock frequency = F_{FS} x 512

Bits 25:20 **MCKDIV[5:0]**: Master clock divider
These bits are set and cleared by software.
000000: Divides by 1 the kernel clock input (sai_x_ker_ck).
Otherwise, the master clock frequency is calculated according to the formula given in *Section 39.4.8*.
These bits have no meaning when the audio block is slave.
They have to be configured when the audio block is disabled.

Bit 19 **NODIV:** No divider
This bit is set and cleared by software.
0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512
1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable
This bit is set and cleared by software.
0: DMA disabled
1: DMA enabled
Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.

Bit 16 **SAIEN**: Audio block enable
This bit is set by software.
To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.
This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.
0: SAI audio block disabled
1: SAI audio block enabled.
Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRIV**: Output drive
This bit is set and cleared by software.
0: Audio block output driven when SAIEN is set
1: Audio block output driven immediately after the setting of this bit.
Note: This bit has to be set before enabling the audio block and after the audio block configuration.

Bit 12 **MONO**: Mono mode
This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to *Section : Mono/stereo mode* for more details.
0: Stereo mode
1: Mono mode.

Bits 11:10 **SYNCEN[1:0]:** Synchronization enable

These bits are set and cleared by software. They must be configured when the audio subblock is disabled.

00: audio subblock in asynchronous mode.

01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode

10: Reserved.

11: Reserved

Note: The audio subblock must be configured as asynchronous when SPDIF mode is enabled.

Bit 9 **CKSTR:** Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.

1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST:** Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC'97 audio protocol since AC'97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

0: Data are transferred with MSB first

1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]:** Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

000: Reserved

001: Reserved

010: 8 bits

011: 10 bits

100: 16 bits

101: 20 bits

110: 24 bits

111: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]:** Protocol configuration

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

00: Free protocol. Free protocol enables to use the powerful config uration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.

01: SPDIF protocol

10: AC'97 protocol

11: Reserved

Bits 1:0 **MODE[1:0]:** SAIx audio block mode

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

00: Master transmitter

01: Master receiver

10: Slave transmitter

11: Slave receiver

Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00). In Master transmitter mode, the audio block starts generating the FS and the clocks immediately.

39.6.10 SAI configuration register 2 (SAI_BCR2)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[1:0]		CPL	MUTECNT[5:0]						MUTE VAL	MUTE	TRIS	F FLUSH	FTH[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **COMP[1:0]**: Companding mode.

These bits are set and cleared by software. The µ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on *CPL bit*.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to *Section : Companding mode* for more details.

00: No companding algorithm

01: Reserved.

10: µ-Law algorithm

11: A-Law algorithm

Note: Companding mode is applicable only when Free protocol mode is selected.

Bit 13 **CPL**: Complement bit.

This bit is set and cleared by software.

It defines the type of complement to be used for companding mode

0: 1’s complement representation.

1: 2’s complement representation.

Note: This bit has effect only when the companding mode is µ-Law algorithm or A-Law algorithm.

Bits 12:7 **MUTECNT[5:0]**: Mute counter.

These bits are set and cleared by software. They are used only in reception mode.

The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.

Refer to *Section : Mute mode* for more details.

Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.

If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.

if the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to *Section : Mute mode* for more details.

0: Bit value 0 is sent during the mute mode.

1: Last values are sent during the mute mode.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 5 **MUTE:** Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to *Section : Mute mode* for more details.

0: No mute mode.

1: Mute mode enabled.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It must be configured when SAI is disabled.

Refer to *Section : Output data line management on an inactive slot* for more details.

0: SD output line is still driven by the SAI when a slot is inactive.

1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH:** FIFO flush.

This bit is set by software. It is always read as 0. This bit must be configured when the SAI is disabled.

0: No FIFO flush.

1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled

Bits 2:0 **FTH[2:0]:** FIFO threshold.

This bit is set and cleared by software.

000: FIFO empty

001: ¼ FIFO

010: ½ FIFO

011: ¾ FIFO

100: FIFO full

101: Reserved

110: Reserved

111: Reserved

39.6.11 SAI frame configuration register (SAI_BFRCR)

Address offset: 0x2C

Reset value: 0x0000 0007

Note: This register has no meaning in AC'97 and SPDIF audio protocol

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF
													rw	rw	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FSALL[6:0]							FRL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **FSOFF**: Frame synchronization offset.
This bit is set and cleared by software. It is meaningless and is not used in AC'97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.
0: FS is asserted on the first bit of the slot 0.
1: FS is asserted one bit before the first bit of the slot 0.

Bit 17 **FSPOL**: Frame synchronization polarity.
This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC'97 or SPDIF audio block configuration.
This bit must be configured when the audio block is disabled.
0: FS is active low (falling edge)
1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.
This bit is set and cleared by software.
0: FS signal is a start frame signal
1: FS signal is a start of frame signal + channel side identification
When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots is dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).
This bit is meaningless and is not used in AC'97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.
These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame
These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.
They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.

These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).

In master mode, if the master clock (available on MCLK_x pin) is used, the frame length must be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

39.6.12 SAI slot register (SAI_BSLOTR)

Address offset: 0x30

Reset value: 0x0000 0000

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOTEN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
				rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:16 **SLOTEN[15:0]**: Slot enable.

These bits are set and cleared by software.

Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).

0: Inactive slot.

1: Active slot.

The slot must be enabled when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.

These bits are set and cleared by software.

The value set in this bitfield represents the number of slots in the audio frame (including the number of inactive slots). The maximum number of slots is 16.

The number of slots must be even if FSDEF bit in the SAI_xFRCR register is set.

The number of slots must be configured when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

0000: Number of slots is 1.

0001: Number of slots is 2.

...

1111: Number of slots is 16.

Bits 7:6 **SLOTSZ[1:0]**: Slot size
This bits is set and cleared by software.
The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.
Refer to *Output data line management on an inactive slot* for information on how to drive SD line.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
01: 16-bit
10: 32-bit
11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset
These bits are set and cleared by software.
The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

39.6.13 SAI interrupt mask register (SAI_BIM)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC'97).
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC'97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.
This bit has a meaning only if the AC'97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode,

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.
It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.
Note: This bit is used only in Free protocol mode and is meaningless in other modes.

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

39.6.14 SAI status register (SAI_BSR)

Address offset: 0x38

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]		
													r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
									r	r	r	r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.
This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).
000: FIFO empty (transmitter and receiver modes)
001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
101: FIFO full (transmitter and receiver modes)
Others: Reserved

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **LFSDET**: Late frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is not present at the right time.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97 or SPDIF mode.
It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register

Bit 5 **AFSDET**: Anticipated frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is detected earlier than expected.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97or SPDIF mode.
It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.
This bit is read only.
0: External AC'97 Codec is ready
1: External AC'97 Codec is not ready
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.
This bit is read only.
0: No FIFO request.
1: FIFO request to read or to write the SAI_xDR.
The request depends on the audio block configuration:
– If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
– If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.
This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.
This bit is read only.
0: Clock configuration is correct
1: Clock configuration does not respect the rule concerning the frame length specification defined in *Section 39.4.6: Frame synchronization* (configuration of FRL[7:0] bit in the SAI_xFRCR register)
This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.
It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.
This bit is read only.
0: No MUTE detection on the SD input line
1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame
This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).
It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets bit CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.
This bit is read only.
0: No overrun/underrun error.
1: Overrun/underrun error detection.
The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.
It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.
This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

39.6.15 SAI clear flag register (SAI_BCLRFR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
									w	w	w		w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC'97or SPDIF mode
Reading this bit always returns the value 0.

Bit 5 **CAFSDET**: Clear anticipated frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC'97or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.
This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

39.6.16 SAI data register (SAI_BDR)

Address offset: 0x40

Reset value: 0x0000 0000

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

Bits 31:0 **DATA[31:0]**: Data
A write to this register loads the FIFO provided the FIFO is not full.
A read from this register empties the FIFO if the FIFO is not empty.

39.6.17 SAI PDM control register (SAI_PDMCR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
						rw	rw			rw	rw				rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CKEN2**: Clock enable of bitstream clock number 2

This bit is set and cleared by software.

0: SAI_CK2 clock disabled

1: SAI_CK2 clock enabled

Note: It is not recommended to configure this bit when PDMEN = 1.

SAI_CK2 might not be available for all SAI instances. Refer to Section 39.3: SAI implementation for details.

Bit 8 **CKEN1**: Clock enable of bitstream clock number 1

This bit is set and cleared by software.

0: SAI_CK1 clock disabled

1: SAI_CK1 clock enabled

Note: It is not recommended to configure this bit when PDMEN = 1.

SAI_CK1 might not be available for all SAI instances. Refer to Section 39.3: SAI implementation for details.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MICNBR[1:0]**: Number of microphones

This bit is set and cleared by software.

00: Configuration with 2 microphones

01: Configuration with 4 microphones

10: Configuration with 6 microphones

11: Configuration with 8 microphones

*Note: It is not recommended to configure this field when PDMEN = 1.**

The complete set of data lines might not be available for all SAI instances. Refer to Section 39.3: SAI implementation for details.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **PDMEN**: PDM enable

This bit is set and cleared by software. This bit enables to control the state of the PDM interface block.

Make sure that the SAI in already operating in TDM master mode before enabling the PDM interface.

0: PDM interface disabled

1: PDM interface enabled

39.6.18 SAI PDM delay register (SAI_PDMDLY)

Address offset: 0x48

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DLYM4R[2:0]			Res.	DLYM4L[2:0]			Res.	DLYM3R[2:0]			Res.	DLYM3L[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DLYM2R[2:0]			Res.	DLYM2L[2:0]			Res.	DLYM1R[2:0]			Res.	DLYM1L[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **DLYM4R[2:0]**: Delay line for second microphone of **pair 4**

This bitfield is set and cleared by software.

000: No delay

001: Delay of 1 T_{SAI_CK} period

010: Delay of 2 T_{SAI_CK} periods

...

111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D4 line is available.Refer to Section 39.3: SAI implementation to check if it is available.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **DLYM4L[2:0]**: Delay line for first microphone of pair 4

This bitfield is set and cleared by software.

000: No delay

001: Delay of 1 T_{SAI_CK} period

010: Delay of 2 T_{SAI_CK} periods

...

111: Delay of 7 of T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D4 line is available.Refer to Section 39.3: SAI implementation to check if it is available.

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **DLYM3R[2:0]**: Delay line for second microphone of pair 3
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.
Note: This bitfield can be used only if D3 line is available.Refer to Section 39.3: SAI implementation to check if it is available.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **DLYM3L[2:0]**: Delay line for first microphone of pair 3
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.
Note: This bitfield can be used only if D3 line is available.Refer to Section 39.3: SAI implementation to check if it is available.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **DLYM2R[2:0]**: Delay line for second microphone of pair 2
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.
Note: This bitfield can be used only if D2 line is available.Refer to Section 39.3: SAI implementation to check if it is available.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **DLYM2L[2:0]**: Delay line for first microphone of pair 2
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.
Note: This bitfield can be used only if D2 line is available.Refer to Section 39.3: SAI implementation to check if it is available.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **DLYM1R[2:0]**: Delay line adjust for second microphone of pair 1
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D1 line is available.Refer to Section 39.3: SAI implementation to check if it is available.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **DLYM1L[2:0]**: Delay line adjust for first microphone of pair 1
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D1 line is available.Refer to Section 39.3: SAI implementation to check if it is available.

39.6.19 SAI register map

Table 383. SAI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	Reserved																																
0x04 or 0x24	SAI_xCR1	Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN	Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
	Reset value					0	0	0	0	0	0	0	0	0		0	0			0	0	0	0	0	0	0	1	0		0	0	0	0
0x08 or 0x28	SAI_xCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP[1:0]		CPL	MUTECN[5:0]						MUTE VAL	MUTE	TRIS	FFLUS	FTH[2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C or 0x2C	SAI_xFRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF	Res.	FSALL[6:0]							FRL[7:0]							
	Reset value														0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x10 or 0x30	SAI_xSLOTR	SLOTEN[15:0]																Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0		0	0	0	0	0

Table 383. SAI register map and reset values (continued)

Offset	Register name	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
0x14 or 0x34	SAI_xIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
	Reset value																										0	0	0	0	0	0	0
0x18 or 0x38	SAI_xSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
	Reset value														0	0	0										0	0	0	1	0	0	0
0x1C or 0x3C	SAI_xCLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
	Reset value																										0	0	0		0	0	0
0x20 or 0x40	SAI_xDR	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	SAI_PDMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
	Reset value																							0	0			0	0				0
0x48	SAI_PDMDLY	Res.	DLYM4R[2:0]			Res.	DLYM4L[2:0]			Res.	DLYM3R[2:0]			Res.	DLYM3L[2:0]			Res.	DLYM2R[2:0]			Res.	DLYM2L[2:0]			Res.	DLYM1R[2:0]			Res.	DLYM1L[2:0]		
	Reset value		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

40 Universal serial bus full-speed host/device interface (USB)

This section applies to STM32WBA25xx devices only.

40.1 USB introduction

The USB peripheral implements an interface between a full-speed USB 2.0 bus and the APB bus.

USB suspend/resume are supported, which permits to stop the device clocks for low-power consumption.

40.2 USB main features

- USB specification version 2.0 full-speed compliant
- Supports both Host and Device modes
- Configurable number of endpoints from 1 to 8
- Dedicated packet buffer memory (SRAM) of 2048 bytes
- Cyclic redundancy check (CRC) generation/checking, Non-return-to-zero Inverted (NRZI) encoding/decoding and bit-stuffing
- Isochronous transfers support
- Double-buffered bulk/isochronous endpoint/channel support
- USB Suspend/Resume operations
- Frame locked clock pulse generation
- USB 2.0 Link Power Management support (Device mode only)
- Battery Charging Specification Revision 1.2 support (Device mode only)
- USB connect / disconnect capability (controllable embedded pull-up resistor on USB_DP line)

40.3 USB implementation

Table 384 describes the USB implementation in the devices.

Table 384. STM32WBA2 USB implementation

USB features(1)	USB
Host mode	X
Number of endpoints	8
Size of dedicated packet buffer memory SRAM	2048 bytes
Dedicated packet buffer memory SRAM access scheme	32 bits
USB 2.0 Link Power Management (LPM) support in device	X

Table 384. STM32WBA2 USB implementation (continued)

USB features(1)	USB
Battery Charging Detection (BCD) support for device	X
Embedded pull-up resistor on USB_DP line	X

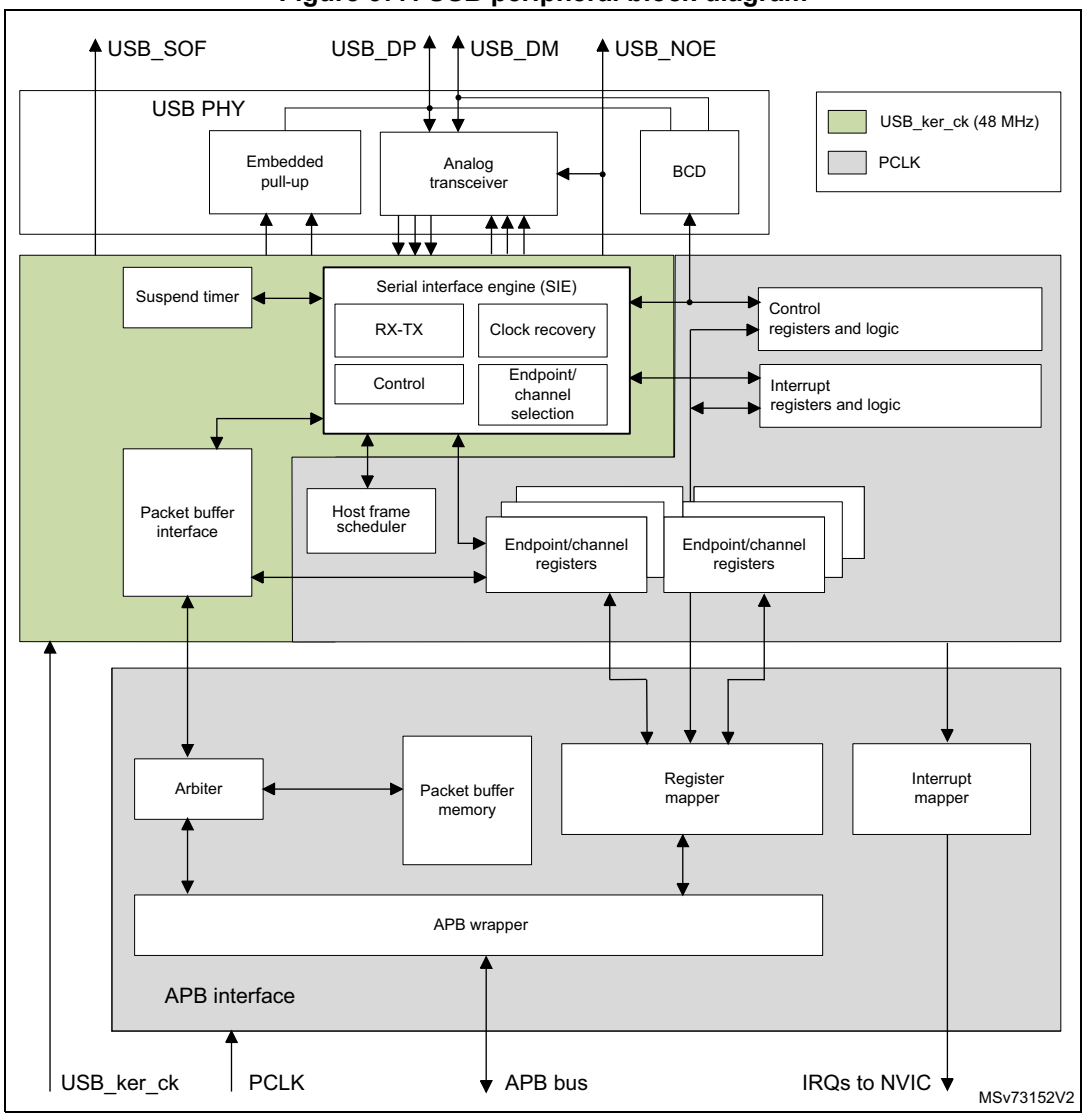
1. X= supported

40.4 USB functional description

40.4.1 USB block diagram

Figure 377 shows the block diagram of the USB peripheral.

Figure 377. USB peripheral block diagram



40.4.2 USB pins and internal signals

Table 385. USB input/output pins

Pin name	Pin type	Description
USB_DP	Digital input/output	D+ line
USB_DM	Digital input/output	D- line
USB_NOE	Digital output	NOE (output enable of data lines)
USB_SOF	Digital output	SOF (start of frame indicator)

40.4.3 USB reset and clocks

A single reset is present on USB. The RCC allows a reset to be forced by software.

There are two clocks:

- PCLK for the APB bus interface and registers.
- USB_ker_ck (48 MHz) for the main protocol logic including notably the serial interface engine (SIE), see USB_ker_ck clock domain in block diagram.

40.4.4 General description and Device mode functionality

The USB peripheral provides a USB-compliant connection between the function implemented by the microcontroller and an external USB function which can be a host PC but also a USB Device. Data transfer between the external USB host or device and the system memory occurs through a dedicated packet buffer memory accessed directly by the USB peripheral. This dedicated memory size is 2048 bytes, and up to 16 mono-directional or 8 bidirectional endpoints can be used. The USB peripheral interfaces with the external USB Host or Device, detecting token packets, handling data transmission/reception, and processing handshake packets as required by the USB standard. Transaction formatting is performed by the hardware, including CRC generation and checking.

Each endpoint/channel is associated with a buffer description block indicating where the endpoint/channel-related memory area is located, how large it is or how many bytes must be transmitted. When a token for a valid function/endpoint pair is recognized by the USB peripheral, the related data transfer (if required and if the endpoint/channel is configured) takes place. The data buffered by the USB peripheral are loaded in an internal 16-bit register and memory access to the dedicated buffer is performed. When all the data have been transferred, if needed, the proper handshake packet over the USB is generated or expected according to the direction of the transfer.

At the end of the transaction, an endpoint/channel-specific interrupt is generated, reading status registers and/or using different interrupt response routines. The microcontroller can determine:

- which endpoint/channel has to be served,
- which type of transaction took place, if errors occurred (bit stuffing, format, CRC, protocol, missing ACK, over/underrun, etc.).

Special support is offered to isochronous transfers and high throughput bulk transfers, implementing a double buffer usage, which permits to always have an available buffer for the USB peripheral while the microcontroller uses the other one.

A special bit THR512 in register USB_ISTR allows notification of 512 bytes being received in (or transmitted from) the buffer. This bit must be used for long ISO packets (from 512 to 1023 bytes) as it facilitates early start or read/write of data. In this way, the first 512 bytes can be handled by software while avoiding use of double buffer mode. This bit works when only one ISO endpoint is configured.

The unit can be placed in low-power mode (SUSPEND mode), by writing in the control register, whenever required. At this time, all static power dissipation is avoided, and the USB clock can be slowed down or stopped. The detection of activity at the USB inputs, while in low-power mode, wakes the device up asynchronously. A special interrupt source can be connected directly to a wake-up line to permit the system to immediately restart the normal clock generation and/or support direct clock start/stop.

Host mode and specific functionality

A single bit, HOST, in register USB_CNTR permits Host mode to be activated. Host mode functionality permits the USB to talk to a remote peripheral. Supported functionality is aligned to Device mode and uses the same register structures to manage the buffers. The same number of endpoints can be supported in Host mode, however in Host mode the terminology “channel” is preferred, as each channel is in reality a combination of the connected device and the endpoint on that device. The basic mechanisms for packet transmission and reception are the same as those supported in Device mode.

When operating in Host mode, the USB is in charge of the bus and in order to do this must issue transaction requests corresponding to active periodic and non-periodic endpoints. A host frame scheduler assures efficient use of the frame. Connection to hubs is supported. Connection to low speed devices is supported, both with a direct connection and through a hub.

Double-buffered mode, as previously described in Device mode, is also supported in Host mode, in both bulk and isochronous channels. The THR512 functionality is also supported (but as in Device mode) only for ISO traffic.

Note: Unlike in Device mode, where there is a detection of battery charging capability (in order to facilitate fast charging), there is no integrated support in Host mode to present battery charging capability (CDP or DCP cases in the standard), the host port is always presented as a default standard data port (SDP).

For LPM (link power management) this feature is not supported in Host mode.

40.4.5 Description of USB blocks used in both Device and Host modes

The USB peripheral implements all the features related to USB interfacing, which include the following blocks:

- USB physical interface (USB PHY): this block is maintaining the electrical interface to an external USB host. It contains the differential analog transceiver itself, controllable embedded pull-up resistor (connected to USB_DP line) and support for battery charging detection (BCD), multiplexed on same USB_DP and USB_DM lines. The output enable control signal of the analog transceiver (active low) is provided externally on USB_NOE. It can be used to drive some activity LED or to provide information about the actual communication direction to some other circuitry.
- Serial interface engine (SIE): the functions of this block include: synchronization pattern recognition, bit-stuffing, CRC generation and checking, PID verification/generation, and handshake evaluation. It must interface with the USB transceivers and uses the virtual buffers provided by the packet buffer interface for

local data storage. This unit also generates signals according to USB peripheral events, such as start of frame (SOF), USB_Reset, data errors etc. and to endpoint related events like end of transmission or correct reception of a packet; these signals are then used to generate interrupts.

- Timer: this block generates a start-of-frame locked clock pulse and detects a global suspend (from the host) when no traffic has been received for 3 ms.
- Packet buffer interface: this block manages the local memory implementing a set of buffers in a flexible way, both for transmission and reception. It can choose the proper buffer according to requests coming from the SIE and locate them in the memory addresses pointed by the endpoint/channel registers. It increments the address after each exchanged byte until the end of packet, keeping track of the number of exchanged bytes and preventing the buffer to overrun the maximum capacity.
- Endpoint/channel-related registers: each endpoint/channel has an associated register containing the endpoint/channel type and its current status. For mono-directional/single-buffer endpoints, a single register can be used to implement two distinct endpoints. The number of registers is 8, allowing up to 16 mono-directional/single-buffer or up to 7 double-buffer endpoints in any combination. For example the USB peripheral can be programmed to have 4 double buffer endpoints and 8 single-buffer/mono-directional endpoints.
- Control registers: these are the registers containing information about the status of the whole USB peripheral and used to force some USB events, such as resume and power-down.
- Interrupt registers: these contain the interrupt masks and a record of the events. They can be used to inquire an interrupt reason, the interrupt status or to clear the status of a pending interrupt.

*Note: * Endpoint/channel 0 is always used for control transfer in single-buffer mode.*

The USB peripheral is connected to the APB bus through an APB interface, containing the following blocks:

- Packet memory: this is the local memory that physically contains the packet buffers. It can be used by the packet buffer interface, which creates the data structure and can be accessed directly by the application software. The size of the packet memory is 2048 bytes, structured as 512 words of 32 bits.
- Arbiter: this block accepts memory requests coming from the APB bus and from the USB interface. It resolves the conflicts by giving priority to APB accesses, while always reserving half of the memory bandwidth to complete all USB transfers. This time-duplex scheme implements a virtual dual-port SRAM that allows memory access, while an USB transaction is happening. Multiword APB transfers of any length are also allowed by this scheme.
- Register mapper: this block collects the various byte-wide and bit-wide registers of the USB peripheral in a structured 32-bit wide word set addressed by the APB.
- APB wrapper: this provides an interface to the APB for the memory and register. It also maps the whole USB peripheral in the APB address space.
- Interrupt mapper: this block is used to select how the possible USB events can generate interrupts and map them to the NVIC.

40.4.6 Description of host frame scheduler (HFS) specific to Host mode

The host frame scheduler is the hardware machine in charge to submit host channel requests on the bus according to the USB priority order and bandwidth access rules.

Host channels are divided in two categories:

- Periodic channels: isochronous and interrupt traffic types. With guaranteed bandwith access.
- Non-periodic channels: bulk and control traffic types. With best effort service.

The host frame scheduler organizes the full-speed frame in 3 sequential windows

- Periodic service window
- Non-periodic service window
- Black security window

At the start of a new frame, the host scheduler:

1. First considers all periodic channels which were active (STAT bits VALID) at the start of frame
2. Executes single round of service of periodic channels, the periodic service window, in hardware priority order from CH#1 to CH#8. For bidirectional channels it executes the OUT direction first
3. When the periodic round is finished, HFS closes the periodic service window and stops servicing periodic traffic even if some periodic channel was re-enabled or some new channel was enabled after the SOF.
4. Starts servicing all non-periodic channels which are currently active (STAT bits VALID) in hardware priority order from CH#1 to CH#8. For bidirectional channels it executes the OUT direction first.
5. Executes multiple round-robin service cycles of non-periodic channels until almost the end of frame
6. Non-periodic traffic can be requested at any time and is serviced by HFS with best effort latency, with the exception of a black security window at the end of the frame where new injected requests are directly postponed to the next frame to avoid babbles. This is also true for pending transactions which have not been serviced ahead of the security window.

40.5 Programming considerations for Device and Host modes

In the following sections, the expected interactions between the USB peripheral and the application program are described, in order to ease application software development.

40.5.1 Generic USB Device programming

This part describes the main tasks required of the application software in order to obtain USB compliant behavior. The actions related to the most general USB events are taken into account and paragraphs are dedicated to the special cases of double-buffered endpoints and isochronous transfers. Apart from system reset, an action is always initiated by the USB peripheral, driven by one of the USB events described below.

40.5.2 System and power-on reset

Upon system and power-on reset, the first operation the application software must perform is to provide all required clock signals to the USB peripheral and subsequently de-assert its reset signal so to be able to access its registers. The whole initialization sequence is hereafter described.

As a first step application software needs to activate register macrocell clock and de-assert macrocell specific reset signal using related control bits provided by device clock management logic.

After that, the analog part of the device related to the USB transceiver must be switched on using the PDWN bit in CNTR register, which requires a special handling. This bit is intended to switch on the internal voltage references that supply the port transceiver. This circuit has a defined startup time ($t_{STARTUP}$ specified in the datasheet) during which the behavior of the USB transceiver is not defined. It is thus necessary to wait this time, after setting the PDWN bit in the CNTR register, before removing the reset condition on the USB part (by clearing the USBRST bit in the CNTR register). Clearing the ISTR register removes any spurious pending interrupt before any other macrocell operation is enabled.

At system reset, the microcontroller must initialize all required registers and the packet buffer description table, to make the USB peripheral able to properly generate interrupts and data transfers. All registers not specific to any endpoint/channel must be initialized according to the needs of application software (choice of enabled interrupts, chosen address of packet buffers, etc.). Then the process continues as for the USB reset case (see further paragraph).

USB bus reset (RST_DCON interrupt) in Device mode

When this event occurs, the USB peripheral is put in the same conditions it is left by the system reset after the initialization described in the previous paragraph: communication is disabled in all endpoint registers (the USB peripheral does not respond to any packet). As a response to the USB reset event, the USB function must be enabled, having as USB address 0, implementing only the default control endpoint (endpoint address is 0 too). This is accomplished by setting the enable function (EF) bit of the USB_DADDR register and initializing the CHEP0R register and its related packet buffers accordingly. During USB enumeration process, the host assigns a unique address to this device, which must be written in the ADD[6:0] bits of the USB_DADDR register, and configures any other necessary endpoint.

When a RST_DCON interrupt is received, the application software is responsible to enable again the default endpoint of USB function 0 within 10 ms from the end of the reset sequence which triggered the interrupt.

USB bus reset in Host mode

In Host mode a bus reset is activated by setting the USBRST bit of the USB_CNTR register. It must subsequently be cleared by software once the minimum active reset time from the standard has been respected.

Structure and usage of packet buffers

Each bidirectional endpoint can receive or transmit data over the bus. The received data is stored in a dedicated memory buffer reserved for that endpoint, while another memory buffer contains the data to be transmitted by the endpoint. Access to this memory is performed by the packet buffer interface block, which delivers a memory access request

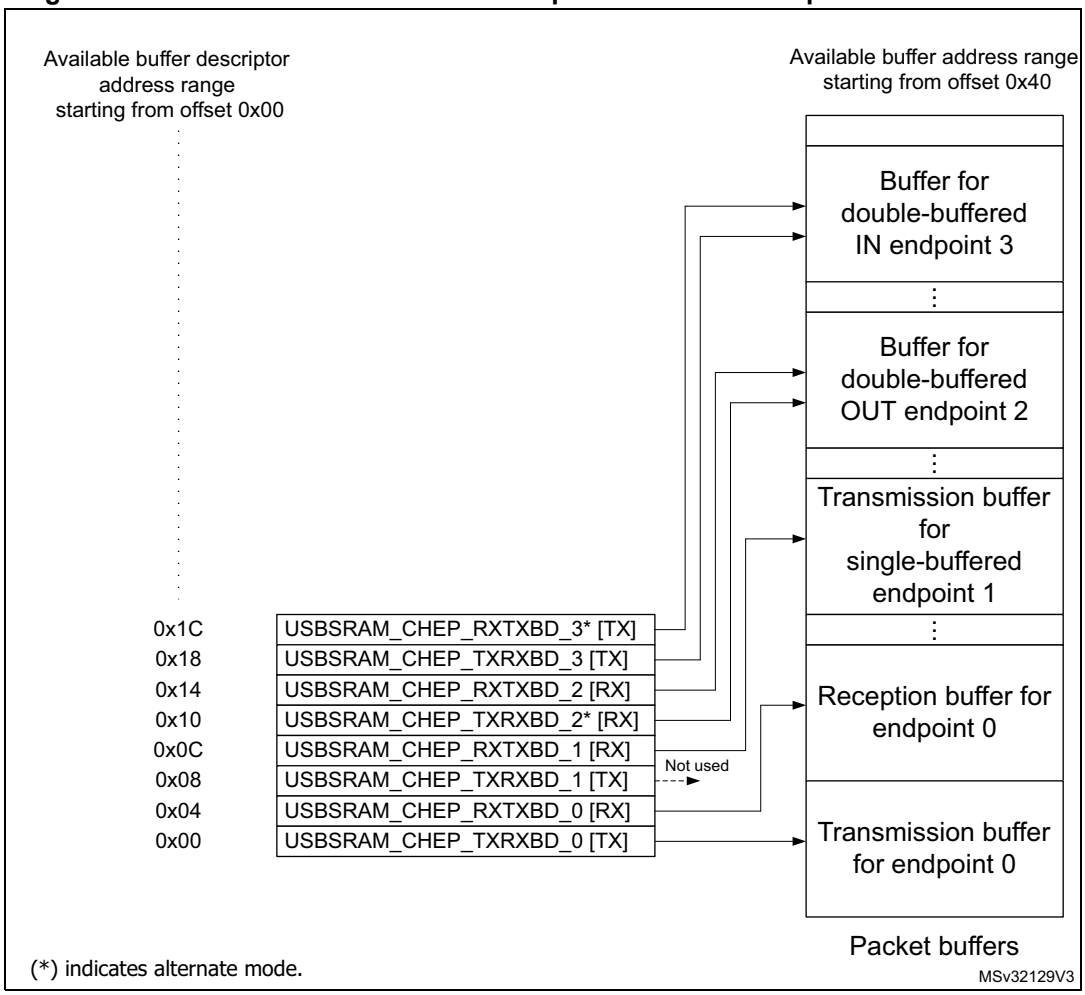
and waits for its acknowledgment. Since the packet buffer memory has also to be accessed by the microcontroller, an arbitration logic takes care of the access conflicts, using half APB cycle for microcontroller access and the remaining half for the USB peripheral access. In this way, both agents can operate as if the packet memory would be a dual-port SRAM, without being aware of any conflict even when the microcontroller is performing back-to-back accesses. The USB peripheral logic uses a dedicated clock. The frequency of this dedicated clock is fixed by the requirements of the USB standard at 48 MHz, and this can be different from the clock used for the interface to the APB bus. Different clock configurations are possible where the APB clock frequency can be higher or lower than the USB peripheral one.

Note: Due to USB data rate and packet memory interface requirements, the APB clock must have a minimum frequency of 12 MHz to avoid data overrun/underrun problems.

Each endpoint is associated with two packet buffers (usually one for transmission and the other one for reception). Buffers can be placed anywhere inside the packet memory because their location and size is specified in a buffer description table, which is also located in the packet memory. Each table entry is associated to an endpoint register and it is composed of two 32-bit words so that table start address must always be aligned to an 8-byte boundary. Buffer descriptor table entries are described in *Section 40.7: USBSRAM registers*. If an endpoint is unidirectional and it is neither an isochronous nor a double-buffered bulk, only one packet buffer is required (the one related to the supported transfer direction). Other table locations related to unsupported transfer directions or unused endpoints, are available to the user. Isochronous and double-buffered bulk endpoints have special handling of packet buffers (Refer to *Section 40.5.5: Isochronous transfers in Device mode* and *Section 40.5.3: Double-buffered endpoints and usage in Device mode* respectively). The relationship between buffer description table entries and packet buffer areas is depicted in *Figure 378*.

For Host mode different sections explain the buffer usage model, notably *Section 40.5.6: Isochronous transfers in Host mode* and *Section 40.5.4: Double buffered channels: usage in Host mode*.

Figure 378. Packet buffer areas with examples of buffer description table locations



Each packet buffer is used either during reception or transmission starting from the bottom. The USB peripheral never changes the contents of memory locations adjacent to the allocated memory buffers; if a packet bigger than the allocated buffer length is received (buffer overrun condition) the data is copied to the memory only up to the last available location.

Endpoint initialization

The first step to initialize an endpoint is to write appropriate values to the ADDRn_TX/ADDRn_RX fields in the CHEP_TXBD_n and CHEP_RXBD_n registers (in SRAM) so that the USB peripheral finds the data to be transmitted already available and the data to be received can be buffered. The UTYPE bits in the USB_CHEPnR register must be set according to the endpoint type, eventually using the EPKIND bit to enable any special required feature. On the transmit side, the endpoint must be enabled using the STATTX bits in the USB_CHEPnR register and COUNTn_TX must be initialized. For reception, STATRX bits must be set to enable reception and COUNTn_RX must be written with the allocated buffer size using the BLSIZE and NUM_BLOCK fields. Unidirectional endpoints, except isochronous and double-buffered bulk endpoints, need to initialize only bits and registers related to the supported direction. Once the transmission and/or reception are enabled, register USB_CHEPnR and locations ADDRn_TX/ADDRn_RX, COUNTn_TX/COUNTn_RX (respectively), must not be modified by the application software, as the hardware can

change their value on the fly. When the data transfer operation is completed, notified by a CTR interrupt event, they can be accessed again to re-enable a new operation.

Data transmission in Device mode (IN packets)

When receiving an IN token packet, if the received address matches a configured and valid endpoint, the USB peripheral accesses the contents of CHEP_TXBD_n (fields ADDRn_TX and COUNTn_TX) inside the buffer descriptor table entry related to the addressed endpoint. The content of these locations is stored in its internal 16-bit registers ADDR and COUNT (not accessible by software). The packet memory is accessed again to read the first byte to be transmitted (refer to *Structure and usage of packet buffers on page 1534*) and the USB peripheral starts sending a DATA0 or DATA1 PID according to USB_CHEPnR bit DTOGTX. When the PID is completed, the first byte, read from buffer memory, is loaded into the output shift register to be transmitted on the USB bus. After the last data byte is transmitted, the computed CRC is sent. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the data packet, according to STATTX bits in the USB_CHEPnR register.

The ADDRn_TX field in the internal register CHEP_TXBD_n is used as a pointer to the current buffer memory location while COUNT is used to count the number of remaining bytes to be transmitted. Each half-word read from the packet buffer memory is transmitted over the USB bus starting from the least significant byte. Transmission buffer memory is read starting from the address pointed by ADDRn_TX for COUNTn_TX/4 words. If a transmitted packet is composed of an odd number of bytes, only the lower half of the last half-word accessed is used.

On receiving the ACK receipt by the host, the USB_CHEPnR register is updated in the following way: DTOGTX bit is toggled, the endpoint is made invalid by setting STATTX = 10 (NAK) and bit VTTX is set. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the IDN and DIR bits in the USB_ISTR register. Servicing of the VTTX event starts, clearing the interrupt bit; the application software then prepares another buffer full of data to be sent, updates the COUNTn_TX table location with the number of byte to be transmitted during the next transfer, and finally sets STATTX to 11 (VALID) to re-enable transmission. While the STATTX bits are equal to 10 (NAK), any IN request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host retries the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second IN transaction addressed to the same endpoint immediately following the one which triggered the CTR interrupt.

Data transmission in Host mode (OUT packets)

Data transmission in Host mode follows the same general principles as Device mode. The main differences are due to the protocol. For example the host initiates the transmission whereas the device responds to the incoming token.

ADDRn_TX must be set to the location in the packet memory reserved for the packet for transmission. The contents of an OUT packet are then written to that address in the packet memory and COUNTn_TX must be updated (when necessary) to indicate the number of bytes in the packet.

DEVADDR must be written for the correct endpoint and then STATTX must be set to 11 (VALID) in order to trigger the transmit. The transmission is then scheduled by the HFS.

After a successful transmission the CTR interrupt (correct transfer) is triggered. By examining IDN and DIR bits, the corresponding channel and direction is understood. On the

indicated channel, the STATTX field now has transitioned to DISABLE. In the case of a NAK being received (when the peripheral is not ready) STATTX is now in NAK. In the case of a STALL response, STATTX is in STALL. In this last case, the bus must be reset.

On receiving the ACK receipt by the device, the USB_CHEPnR register is updated in the following way: DTOGTX bit is toggled.

An error condition is signaled via the bits VTTX and ERR_TX if one of the following conditions occurs:

- No handshake being received in time
- False EOP
- Bit stuffing error
- Invalid handshake PID

Data reception in Device mode (OUT and SETUP packets)

These two tokens are handled by the USB peripheral more or less in the same way; the differences in the handling of SETUP packets are detailed in the following paragraph about control transfers. When receiving an OUT/SETUP PID, if the address matches a valid endpoint, the USB peripheral accesses the contents of the ADDRn_RX and COUNTn_RX fields inside the buffer descriptor table entry related to the addressed endpoint. The content of the ADDRn_RX field is stored directly in its internal register ADDR. Internal register COUNT is now reset and the values of BLSIZE and NUM_BLOCK bit fields, which are read within USBSRAM_CHEP_RXBD_n content, are used to initialize BUF_COUNT, an internal 16-bit counter, which is used to check the buffer overrun condition (all these internal registers are not accessible by software). Data bytes subsequently received by the USB peripheral are packed in half-words (the first byte received is stored as least significant byte) and then transferred to the packet buffer starting from the address contained in the internal ADDR register while BUF_COUNT is decremented and COUNT is incremented at each byte transfer. When the end of DATA packet is detected, the correctness of the received CRC is tested and only if no errors occurred during the reception, an ACK handshake packet is sent back to the transmitting host.

In case of wrong CRC or other kinds of errors (bit-stuff violations, frame errors, etc.), data bytes are still copied in the packet memory buffer, at least until the error detection point, but the ACK packet is not sent and the ERR bit in USB_ISTR register is set. However, there is usually no software action required in this case: the USB peripheral recovers from reception errors and remains ready for the next transaction to come. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the ACK, according to bits STATRX in the USB_CHEPnR register, and no data is written in the reception memory buffers.

Reception memory buffer locations are written starting from the address contained in the ADDRn_RX for a number of bytes corresponding to the received data packet length, or up to the last allocated memory location, as defined by BLSIZE and NUM_BLOCK, whichever comes first. In this way, the USB peripheral never writes beyond the end of the allocated reception memory buffer area. If the length of the data packet payload (actual number of bytes used by the application) is greater than the allocated buffer, the USB peripheral detects a buffer overrun condition. In this case, a STALL handshake is sent instead of the usual ACK to notify the problem to the host, no interrupt is generated and the transaction is considered failed.

When the transaction is completed correctly, by sending the ACK handshake packet, the internal COUNT register is copied back in the COUNTn_RX location inside the buffer

description table entry, leaving unaffected BLSIZE and NUM_BLOCK fields, which normally do not require to be re-written, and the USB_CHEPnR register is updated in the following way: DTOGRX bit is toggled, the endpoint is made invalid by setting STATRX = 10 (NAK) and bit VTRX is set. If the transaction has failed due to errors or buffer overrun condition, none of the previously listed actions take place. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the IDN and DIR bits in the USB_ISTR register. The VTRX event is serviced by first determining the transaction type (SETUP bit in the USB_CHEPnR register); the application software must clear the interrupt flag bit and get the number of received bytes reading the COUNTn_RX location inside the buffer description table entry related to the endpoint being processed. After the received data is processed, the application software must set the STATRX bits to 11 (VALID) in the USB_CHEPnR, enabling further transactions. While the STATRX bits are equal to 10 (NAK), any OUT request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host retries the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second OUT transaction addressed to the same endpoint following immediately the one which triggered the CTR interrupt.

Data reception in Host mode (IN packets)

Data reception in Host mode follows the same general principles as Device mode. The main differences are again due to the protocol. In the device, data can be received or not, depending on readiness after previous operations, whereas the host only requests receive data when it is ready and able to store them.

ADDRn_TX must be set to the location in the packet memory reserved for the packet for transmission. The contents received in the data phase response to the IN token packet are then written to that address in the packet memory and COUNTn_TX gets updated by hardware during this process to indicate the number of bytes in the packet.

DEVADDR must be written for the correct endpoint and then STATRX must be set to VALID in order to trigger the reception. The reception is then scheduled by the HFS.

After a successful reception the interrupt CTR (correct transfer) is triggered. By examining IDN and DIR bits, the corresponding channel and direction is understood. On the indicated channel, the STATRX field now has transitioned to DISABLE. In the case of a NAK being received (when the peripheral is not ready) STATRX now is in NAK. In the case of a STALL response, STATRX is in STALL. In this last case, the bus must be reset. During an IN packet an error condition is signaled via the bits VTRX and ERR_RX if one of the following conditions occurs:

- False EOP
- Bit stuffing error
- Wrong CRC

Control transfers in Device mode

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only and are very similar to OUT ones (data reception) except that the values of DTOGTX and DTOGRX bits of the addressed endpoint registers are set to 1 and 0 respectively, to initialize the control transfer, and both STATTX and STATRX are set to 10 (NAK) to let software decide if subsequent transactions must be IN or OUT depending on the SETUP contents. A control endpoint must check SETUP bit in the USB_CHEPnR register at each VTRX event to distinguish normal

OUT transactions from SETUP ones. A USB Device can determine the number and direction of data stages by interpreting the data transferred in the SETUP stage, and is required to STALL the transaction in the case of errors. To do so, at all data stages before the last, the unused direction must be set to STALL, so that, if the host reverses the transfer direction too soon, it gets a STALL as a status stage.

While enabling the last data stage, the opposite direction must be set to NAK, so that, if the host reverses the transfer direction (to perform the status stage) immediately, it is kept waiting for the completion of the control operation. If the control operation completes successfully, the software changes NAK to VALID, otherwise to STALL. At the same time, if the status stage is an OUT, the STATUS_OUT (EPKIND in the USB_CHEPnR register) bit must be set, so that an error is generated if a status transaction is performed with non-zero data. When the status transaction is serviced, the application clears the STATUS_OUT bit and sets STATRX to VALID (to accept a new command) and STATTX to NAK (to delay a possible status stage immediately following the next setup).

Since the USB specification states that a SETUP packet cannot be answered with a handshake different from ACK, eventually aborting a previously issued command to start the new one, the USB logic does not permit a control endpoint to answer with a NAK or STALL packet to a SETUP token received from the host.

When the STATRX bits are set to 01 (STALL) or 10 (NAK) and a SETUP token is received, the USB accepts the data, performing the required data transfers and sends back an ACK handshake. If that endpoint has a previously issued VTRX request not yet acknowledged by the application (for example VTRX bit is still set from a previously completed reception), the USB discards the SETUP transaction and does not answer with any handshake packet regardless of its state, simulating a reception error and forcing the host to send the SETUP token again. This is done to avoid losing the notification of a SETUP transaction addressed to the same endpoint immediately following the transaction, which triggered the VTRX interrupt.

Control transfers in Host mode

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only. A control endpoint must set the SETUP bit in the USB_CHEPnR register. The values of DTOGTX and DTOGRX bits of the addressed endpoint registers are set to 0. Depending on whether it is a control write or control read then STATTX or STATRX are set to 11 (ACTIVE) in order to trigger the control transfer via the host frame scheduler.

On receiving a CTR interrupt the channel (device address and endpoint) can be determined by examining IDN and DIR bits. Devices are expected to NAK every control unless the packet is corrupted in which case they do not acknowledge. The situation is reflected in the value of STATTX.

In the case of an error condition the ERR bit gets set. One possible case is where a CRC error is seen at the device, in this case no ACK is returned to the host. The host sees no ACK and after an appropriate delay this generates a timeout error with ERR_TX set (which can generate an interrupt).

40.5.3 Double-buffered endpoints and usage in Device mode

All different endpoint types defined by the USB standard represent different traffic models, and describe the typical requirements of different kind of data transfer operations. When large portions of data are to be transferred between the host PC and the USB function, the bulk endpoint type is the most suited model. This is because the host schedules bulk transactions so as to fill all the available bandwidth in the frame, maximizing the actual transfer rate as long as the USB function is ready to handle a bulk transaction addressed to it. If the USB function is still busy with the previous transaction when the next one arrives, it answers with a NAK handshake and the host PC issues the same transaction again until the USB function is ready to handle it, reducing the actual transfer rate due to the bandwidth occupied by re-transmissions. For this reason, a dedicated feature called ‘double-buffering’ can be used with bulk endpoints.

When ‘double-buffering’ is activated, data toggle sequencing is used to select, which buffer is to be used by the USB peripheral to perform the required data transfers, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to always have a complete buffer to be used by the application, while the USB peripheral fills the other one. For example, during an OUT transaction directed to a ‘reception’ double-buffered bulk endpoint, while one buffer is being filled with new data coming from the USB host, the other one is available for the microcontroller software usage (the same would happen with a ‘transmission’ double-buffered bulk endpoint and an IN transaction).

Since the swapped buffer management requires the usage of all 4 buffer description table locations hosting the address pointer and the length of the allocated memory buffers, the USB_CHEPnR registers used to implement double-buffered bulk endpoints are forced to be used as unidirectional ones. Therefore, only one STAT bit pair must be set at a value different from 00 (DISABLED): STATRX if the double-buffered bulk endpoint is enabled for reception, STATTX if the double-buffered bulk endpoint is enabled for transmission. In case it is required to have double-buffered bulk endpoints enabled both for reception and transmission, two USB_CHEPnR registers must be used.

To exploit the double-buffering feature and reach the highest possible transfer rate, the endpoint flow control structure, described in previous chapters, has to be modified, in order to switch the endpoint status to NAK only when a buffer conflict occurs between the USB peripheral and application software, instead of doing it at the end of each successful transaction. The memory buffer which is currently being used by the USB peripheral is defined by the DTOG bit related to the endpoint direction: DTOGRX (bit 14 of USB_CHEPnR register) for ‘reception’ double-buffered bulk endpoints or DTOGTX (bit 6 of USB_CHEPnR register) for ‘transmission’ double-buffered bulk endpoints. To implement the new flow control scheme, the USB peripheral must know which packet buffer is currently in use by the application software, so to be aware of any conflict. Since in the USB_CHEPnR register, there are two DTOG bits but only one is used by USB peripheral for data and buffer sequencing (due to the unidirectional constraint required by double-buffering feature) the other one can be used by the application software to show which buffer it is currently using. This new buffer flag is called SW_BUF. In the following table the correspondence between USB_CHEPnR register bits and DTOG/SW_BUF definition is explained, for the cases of ‘transmission’ and ‘reception’ double-buffered bulk endpoints.

Table 386. Double-buffering buffer flag definition

Buffer flag	'Transmission' endpoint	'Reception' endpoint
DTOG	DTOGTX (USB_CHEPnR bit 6)	DTOGRX (USB_CHEPnR bit 14)
SW_BUF	USB_CHEPnR bit 14	USB_CHEPnR bit 6

The memory buffer which is currently being used by the USB peripheral is defined by DTOG buffer flag, while the buffer currently in use by application software is identified by SW_BUF buffer flag. The relationship between the buffer flag value and the used packet buffer is the same in both cases, and it is listed in the following table.

Table 387. Bulk double-buffering memory buffers usage (Device mode)

Endpoint type	DTOG	SW_BUF	Packet buffer used by USB peripheral	Packet buffer used by Application Software
Transmit (IN)	0	1	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations
	1	0	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	0	0	None (1)	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	1	1	None (1)	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (OUT)	0	1	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	0	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	0	0	None (1)	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	1	None (1)	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.

1. Endpoint in NAK Status.

Double-buffering feature for a bulk endpoint is activated by performing the two following actions:

- Writing UTYPE bit field at 00 in its USB_CHEPnR register, to define the endpoint as a bulk
- Setting EPKIND bit at 1 (DBL_BUF), in the same register.

The application software is responsible for DTOG and SW_BUF bits initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. The end of the first transaction occurring after having set DBL_BUF, triggers the special flow control of double-buffered bulk endpoints, which is used for all other transactions addressed to this endpoint until DBL_BUF remain set. At the end of each transaction the VTRX or VTTX bit of the addressed endpoint USB_CHEPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_CHEPnR register is hardware toggled making the USB peripheral buffer swapping completely software independent. Unlike common transactions, and the first one after DBL_BUF setting, STAT bit pair is not affected by the transaction termination and its value remains 11 (VALID). However, as the token packet of a new transaction is received, the actual endpoint status is masked as 10 (NAK) when a buffer conflict between the USB peripheral and the application software is detected (this condition is identified by DTOG and SW_BUF having the same value, see *Table 387*). The application software responds to the CTR event notification by clearing the interrupt flag and starting any required handling of the completed transaction. When the application packet buffer usage is over, the software toggles the SW_BUF bit, writing 1 to it, to notify the USB peripheral about the availability of that buffer. In this way, the number of NAKed transactions is limited only by the application elaboration time of a transaction data: if the elaboration time is shorter than the time required to complete a transaction on the USB bus, no re-transmissions due to flow control takes place and the actual transfer rate is limited only by the host PC.

The application software can always override the special flow control implemented for double-buffered bulk endpoints, writing an explicit status different from 11 (VALID) into the STAT bit pair of the related USB_CHEPnR register. In this case, the USB peripheral always uses the programmed endpoint status, regardless of the buffer usage condition.

40.5.4 Double buffered channels: usage in Host mode

In Host mode the underlying transmit and receive methods for double buffered channels are the same as those described for Device mode.

Similar to the Device mode table, a new table below *Table 388: Bulk double-buffering memory buffers usage (Host mode)* shows the programming settings for OUT and IN tokens.

Table 388. Bulk double-buffering memory buffers usage (Host mode)

Endpoint type	DTOG	SW_BUF	Packet buffer used by USB peripheral	Packet buffer used by Application Software
Transmit (OUT)	0	1	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations
	1	0	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	0	0	None (1)	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	1	1	None (1)	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (IN)	0	1	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	0	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	0	0	None (1)	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	1	None (1)	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.

1. Endpoint in NAK Status.

40.5.5 Isochronous transfers in Device mode

The USB standard supports full speed peripherals requiring a fixed and accurate data production/consume frequency, defining this kind of traffic as ‘isochronous’. Typical examples of this data are: audio samples, compressed video streams, and in general any sort of sampled data having strict requirements for the accuracy of delivered frequency. When an endpoint is defined to be ‘isochronous’ during the enumeration phase, the host allocates in the frame the required bandwidth and delivers exactly one IN or OUT packet each frame, depending on endpoint direction. To limit the bandwidth requirements, no re-transmission of failed transactions is possible for isochronous traffic; this leads to the fact that an isochronous transaction does not have a handshake phase and no ACK packet is expected or sent after the data packet. For the same reason, isochronous transfers do not support data toggle sequencing and always use DATA0 PID to start any data packet.

The isochronous behavior for an endpoint is selected by setting the UTYPE bits at 10 in its USB_CHEPnR register; since there is no handshake phase the only legal values for the STATRX/STATTX bit pairs are 00 (DISABLED) and 11 (VALID), any other value produces results not compliant to USB standard. Isochronous endpoints implement double-buffering

to ease application software development, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to have always a complete buffer to be used by the application, while the USB peripheral fills the other.

The memory buffer which is currently used by the USB peripheral is defined by the DTOG bit related to the endpoint direction (DTOGRX for ‘reception’ isochronous endpoints, DTOGTX for ‘transmission’ isochronous endpoints, both in the related USB_CHEPnR register) according to *Table 389*.

Table 389. Isochronous memory buffers usage

Endpoint Type	DTOG bit value	Packet buffer used by the USB peripheral	Packet buffer used by the application software
Transmit (IN)	0	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations
	1	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (OUT)	0	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.

As it happens with double-buffered bulk endpoints, the USB_CHEPnR registers used to implement isochronous endpoints are forced to be used as unidirectional ones. In case it is required to have isochronous endpoints enabled both for reception and transmission, two USB_CHEPnR registers must be used.

The application software is responsible for the DTOG bit initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. At the end of each transaction, the VTRX or VTTX bit of the addressed endpoint USB_CHEPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_CHEPnR register is hardware toggled making buffer swapping completely software independent. STAT bit pair is not affected by transaction completion; since no flow control is possible for isochronous transfers due to the lack of handshake phase, the endpoint remains always 11 (VALID). CRC errors or buffer-overrun conditions occurring during isochronous OUT transfers are anyway considered as correct transactions and they always trigger a VTRX event. However, CRC errors set the ERR bit in the USB_ISTR register anyway, in order to notify the software of the possible data corruption.

40.5.6 Isochronous transfers in Host mode

From the host point of view isochronous packets are issued or requested one by frame by the host frame scheduler. There is no NAK/ACK protocol and no resend of data or token.

The mechanism is based on a table very similar to that for Device mode. See *Table 390* to understand the relationship between the DTOG bit buffers and the buffer usage.

Table 390. Isochronous memory buffers usage

Endpoint Type	DTOG bit value	Packet buffer used by the USB peripheral	Packet buffer used by the application software
Transmit (OUT)	0	USBSRAM_CHEP_TXRXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations.	USBSRAM_CHEP_RXTXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations
	1	USBSRAM_CHEP_RXTXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations	USBSRAM_CHEP_TXRXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations.
Receive (IN)	0	USBSRAM_CHEP_RXTXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.	USBSRAM_CHEP_TXRXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.
	1	USBSRAM_CHEP_TXRXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations	USBSRAM_CHEP_RXTXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.

The isochronous behavior for an endpoint is selected by setting the UTYPE bits at 10 in its USB_CHEPnR register; since there is no handshake phase the only legal values for the STATRX/STATTX bit pairs are 00 (DISABLED) and 11 (VALID),

Just as in Device mode, the mechanism allows automatic toggle of the DTOG bit. Note that in Host mode, at the same time as this toggle, the STATTX or STATRX of the completed buffer is automatically set to DISABLED, permitting the future buffer to be accessed before re-enabling it by setting it to 11 (VALID).

40.5.7 Suspend/resume events

The USB standard defines a special peripheral state, called SUSPEND, in which the average current drawn from the USB bus must not be greater than 2.5 mA. This requirement is of fundamental importance for bus-powered devices, while self-powered devices are not required to comply to this strict power consumption constraint. In suspend mode, the host PC sends the notification by not sending any traffic on the USB bus for more than 3 ms: since a SOF packet must be sent every 1 ms during normal operations, the USB peripheral detects the lack of 3 consecutive SOF packets as a suspend request from the host PC and set the SUSP bit to 1 in USB_ISTR register, causing an interrupt if enabled. Once the device is suspended, its normal operation can be restored by a so called RESUME sequence, which can be started from the host PC or directly from the peripheral itself, but it is always terminated by the host PC. The suspended USB peripheral must be anyway able to detect a RESET sequence, reacting to this event as a normal USB reset event.

The actual procedure used to suspend the USB peripheral is device dependent since according to the device composition, different actions may be required to reduce the total consumption.

A brief description of a typical suspend procedure is provided below, focused on the USB-related aspects of the application software routine responding to the SUSP notification of the USB peripheral:

1. Set the SUSPEN bit in the USB_CNTR register to 1. This action activates the suspend mode within the USB peripheral. As soon as the suspend mode is activated, the check on SOF reception is disabled to avoid any further SUSP interrupts being issued while the USB is suspended.
2. Remove or reduce any static power consumption in blocks different from the USB peripheral.
3. Set SUSPRDY bit in USB_CNTR register to 1 to remove static power consumption in the analog USB transceivers but keeping them able to detect resume activity.
4. Optionally turn off external oscillator and device PLL to stop any activity inside the device.

When an USB event occurs while the device is in SUSPEND mode, the RESUME procedure must be invoked to restore nominal clocks and regain normal USB behavior. Particular care must be taken to ensure that this process does not take more than 10 ms when the wakening event is an USB reset sequence (see “Universal Serial Bus Specification” for more details). The start of a resume or reset sequence, while the USB peripheral is suspended, clears the SUSPRDY bit in USB_CNTR register asynchronously. Even if this event can trigger a WKUP interrupt if enabled, the use of an interrupt response routine must be carefully evaluated because of the long latency due to system clock restart; to have the shorter latency before re-activating the nominal clock it is suggested to put the resume procedure just after the end of the suspend one, so its code is immediately executed as soon as the system clock restarts. To prevent ESD discharges or any other kind of noise from waking-up the system (the exit from suspend mode is an asynchronous event), a suitable analog filter on data line status is activated during suspend; the filter width is about 70 ns.

The following is a list of actions a resume procedure must address:

1. Optionally turn on external oscillator and/or device PLL.
2. Clear SUSPEN bit of USB_CNTR register.
3. If the resume triggering event has to be identified, bits RXDP and RXDM in the USB_FNR register can be used according to *Table 391*, which also lists the intended software action in all the cases. If required, the end of resume or reset sequence can be detected monitoring the status of the above mentioned bits by checking when they reach the “10” configuration, which represent the idle bus state; moreover at the end of a reset sequence the RST_DCON bit in USB_ISTR register is set to 1, issuing an interrupt if enabled, which must be handled as usual.

Table 391. Resume event detection

[RXDP,RXDM] status	Wake-up event	Required resume software action
“00”	Root reset	None
“10”	None (noise on bus)	Go back in Suspend mode
“01”	Root resume	None
“11”	Not allowed (noise on bus)	Go back in Suspend mode

A device may require to exit from suspend mode as an answer to particular events not directly related to the USB protocol (for example a mouse movement wakes up the whole system). In this case, the resume sequence can be started by setting the L2RES bit in the USB_CNTR register to 1 and resetting it to 0 after an interval between 1 ms and 15 ms (this interval can be timed using ESOF interrupts, occurring with a 1 ms period when the system clock is running at nominal frequency). Once the L2RES bit is clear, the resume sequence is completed by the host PC and its end can be monitored again using the RXDP and RXDM bits in the USB_FNR register.

Note: The L2RES bit must be anyway used only after the USB peripheral has been put in suspend mode, setting the SUSPEN bit in USB_CNTR register to 1.

Suspend and resume in Host mode

The basics of the suspend and resume mechanism has been described in the previous section.

From the host stand-point, suspend is entered by writing the SUSPEN bit in USB_CNTR. When suspend entry is confirmed, SUSPRDY (also in USB_CNTR) is set.

Once in suspend, and when the application want to resume the bus, this can be done by setting the L2RES bit in USB_CNTR to 1.

Below in *Table 392*, the different actions recommended after a wake-up event are indicated. According to the different line states after a wake-up event, the interpretation of the event and the suggested behavior are shown. Note that, this table here is somewhat expanded when compared to the previously shown device table, as the host may encounter both full speed and low speed devices which use different line states for both suspend and resume.

Table 392. Resume event detection for host

[RXDP,RXDM] status	Wake-up event	Required resume software action
“00”	Not allowed (noise on bus)	Go back in Suspend mode
“10”	Full speed capable device: Not allowed (noise on bus) Low speed device: Device remote wake-up resume	None
“01”	Full speed capable device: Device remote wake-up resume Low speed device: Not allowed (noise on bus)	None
“11”	Not allowed (noise on bus)	Go back in Suspend mode

40.6 USB registers

The USB peripheral registers can be divided into the following groups:

- Common registers: interrupt and control registers. These registers affect the general behavior of the USB peripheral defining operating mode, interrupt handling, device address and giving access to the current frame number updated by the host PC.
  - USB_CNTR
  - USB_ISTR
  - USB_FNR
  - USB_DADDR
  - USB_LPMCSR
  - USB_BCDR
- Endpoint/channel registers: endpoint/channel configuration and status
  - USB_CHEPnR

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

40.6.1 USB control register (USB_CNTR)

Address offset: 0x40

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HOST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISCM	THR512M
rw														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RST_DCONM	SOFM	ESOFM	L1REQM	Res.	L1RES	L2RES	SUSPEN	SUSPRDY	PDWN	USBRST
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	r	rw	rw

Bit 31 **HOST:** HOST mode

HOST bit selects betweens host or device USB mode of operation. It must be set before enabling the USB peripheral by the function enable bit.

0: USB Device function

1: USB host function

Bits 30:18 Reserved, must be kept at reset value.

Bit 17 **DDISCM**: Device disconnection mask

– Host mode

0: Device disconnection interrupt disabled

1: Device disconnection interrupt enabled

Bit 16 **THR512M**: 512 byte threshold interrupt mask

0: 512 byte threshold interrupt disabled

1: 512 byte threshold interrupt enabled

Bit 15 **CTRM:** Correct transfer interrupt mask

0: Correct transfer (CTR) interrupt disabled.

1: CTR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 14 **PMAOVRM:** Packet memory area over / underrun interrupt mask

0: PMAOVR interrupt disabled.

1: PMAOVR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 13 **ERRM:** Error interrupt mask

0: ERR interrupt disabled.

1: ERR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 12 **WKUPM:** Wake-up interrupt mask

0: WKUP interrupt disabled.

1: WKUP interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 11 **SUSPM:** Suspend mode interrupt mask

0: Suspend mode request (SUSP) interrupt disabled.

1: SUSP interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 10 **RST_DCONM:** USB reset request (Device mode) or device connect/disconnect (Host mode) interrupt mask
0: RESET interrupt disabled.
1: RESET interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 9 **SOFM:** Start of frame interrupt mask
0: SOF interrupt disabled.
1: SOF interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 8 **ESOFM:** Expected start of frame interrupt mask
0: Expected start of frame (ESOF) interrupt disabled.
1: ESOF interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 7 **L1REQM:** LPM L1 state request interrupt mask
0: LPM L1 state request (L1REQ) interrupt disabled.
1: L1REQ interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **L1RES:** L1 remote wake-up / resume driver
– Device mode
Software sets this bit to send a LPM L1 50 µs remote wake-up signaling to the host. After the signaling ends, this bit is cleared by hardware.
0: No effect
1: Send 50 µs remote-wake-up signaling to host

Bit 4 **L2RES:** L2 remote wake-up / resume driver
– Device mode
The microcontroller can set this bit to send remote wake-up signaling to the host. It must be activated, according to USB specifications, for no less than 1 ms and no more than 15 ms after which the host PC is ready to drive the resume sequence up to its end.
– Host mode
Software sets this bit to send resume signaling to the device.
Software clears this bit to send end of resume to device and restart SOF generation.
In the context of remote wake up, this bit is to be set following the WAKEUP interrupt.
0: No effect
1: Send L2 resume signaling to device

Bit 3 **SUSPEN:** Suspend state enable

– Condition: Device mode

Software can set this bit when the SUSP interrupt is received, which is issued when no traffic is received by the USB peripheral for 3 ms. Software can also set this bit when the L1REQ interrupt is received with positive acknowledge sent.

As soon as the suspend state is propagated internally all device activity is stopped, USB clock is gated, USB transceiver is set into low power mode and the SUSPRDY bit is set by hardware. In the case that device application wants to pursue more aggressive power saving by stopping the USB clock source and by moving the microcontroller to stop mode, as in the case of bus powered device application, it must first wait few cycles to see the SUSPRDY = 1 acknowledge the suspend request.

This bit is cleared by hardware simultaneous with the WAKEUP flag set.

0: No effect

1: Enter L1/L2 suspend

– Condition: Host mode

Software can set this bit when host application has nothing scheduled for the next frames and wants to enter long term power saving. When set, it stops immediately SOF generation and any other host activity, gates the USB clock and sets the transceiver in low power mode. If any USB transaction is on-going at the time SUSPEN is set, suspend is entered at the end of the current transaction.

As soon as suspend state is propagated internally and gets effective the SUSPRDY bit is set. In the case that host application wants to pursue more aggressive power saving by stopping the USB clock source and by moving the micro-controller to STOP mode, it must first wait few cycles to see SUSPRDY=1 acknowledge to the suspend request.

This bit is cleared by hardware simultaneous with the WAKEUP flag set.

0: No effect

1: Enter L1/L2 suspend

Bit 2 **SUSPRDY:** Suspend state effective

This bit is set by hardware as soon as the suspend state entered through the SUSPEN control gets internally effective. In this state USB activity is suspended, USB clock is gated, transceiver is set in low power mode by disabling the differential receiver. Only asynchronous wake-up logic and single ended receiver is kept alive to detect remote wake-up or resume events.

Software must poll this bit to confirm it to be set before any STOP mode entry.

This bit is cleared by hardware simultaneously to the WAKEUP flag being set.

0: Normal operation

1: Suspend state

Bit 1 **PDWN:** Power down

This bit is used to completely switch off all USB-related analog parts if it is required to completely disable the USB peripheral for any reason. When this bit is set, the USB peripheral is disconnected from the transceivers and it cannot be used.
0: Exit power down
1: Enter power down mode

Bit 0 **USBRST:** USB Reset

– Condition: Device mode

Software can set this bit to reset the USB core, exactly as it happens when receiving a RESET signaling on the USB.The USB peripheral, in response to a RESET, resets its internal protocol state machine. Reception and transmission are disabled until the RST_DCON bit is cleared. All configuration registers do not reset: the microcontroller must explicitly clear these registers (this is to ensure that the RST_DCON interrupt can be safely delivered, and any transaction immediately followed by a RESET can be completed). The function address and endpoint registers are reset by an USB reset event.
0: No effect
1: USB core is under reset

– Condition: Host mode

Software sets this bit to drive USB reset state on the bus and initialize the device. USB reset terminates as soon as this bit is cleared by software.
0: No effect
1: USB reset driven

40.6.2 USB interrupt status register (USB_ISTR)

Address offset: 0x44

Reset value: 0x0000 0000

This register contains the status of all the interrupt sources permitting application software to determine which events caused an interrupt request.

The upper part of this register contains single bits, each of them representing a specific event. These bits are set by the hardware when the related event occurs; if the corresponding bit in the USB_CNTR register is set, a generic interrupt request is generated. The interrupt routine, examining each bit, performs all necessary actions, and finally it clears the serviced bits. If any of them is not cleared, the interrupt is considered to be still pending, and the interrupt line is kept high again. If several bits are set simultaneously, only a single interrupt is generated.

Endpoint/channel transaction completion can be handled in a different way to reduce interrupt response latency. The CTR bit is set by the hardware as soon as an endpoint/channel successfully completes a transaction, generating a generic interrupt request if the corresponding bit in USB_CNTR is set. An endpoint/channel dedicated interrupt condition is activated independently from the CTRM bit in the USB_CNTR register. Both interrupt conditions remain active until software clears the pending bit in the corresponding USB_CHEPnR register (the CTR bit is actually a read only bit). For endpoint-/channel-related interrupts, the software can use the direction of transaction (DIR) and IDN read-only bits to identify which endpoint/channel made the last interrupt request and called the corresponding interrupt service routine.

The user can choose the relative priority of simultaneously pending USB_ISTR events by specifying the order in which software checks USB_ISTR bits in an interrupt service routine. Only the bits related to events, which are serviced, are cleared. At the end of the service routine, another interrupt is requested, to service the remaining conditions.

To avoid spurious clearing of some bits, it is recommended to clear them with a load instruction where all bits which must not be altered are written with 1, and all bits to be cleared are written with 0 (these bits can only be cleared by software). Read-modify-write cycles must be avoided because between the read and the write operations another bit can be set by the hardware and the next write clears it before the device has the time to service the event.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LS_ DCON	DCON_ STAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISC	THR 512
	r	r												rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR	PMA OVR	ERR	WKUP	SUSP	RST_ DCON	SOF	ESOF	L1REQ	Res.	Res.	DIR	IDN[3:0]			
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0			r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **LS_DCON**: Low speed device connected

– Host mode:

This bit is set by hardware when an LS device connection is detected. Device connection is signaled after LS J-state is sampled for 22 consecutive cycles of the USB clock (48 MHz) from the unconnected state.

Bit 29 **DCON_STAT**: Device connection status

– Host mode:

This bit contains information about device connection status. It is set by hardware when a LS/FS device is attached to the host while it is reset when the device is disconnected.

0: No device connected

1: FS or LS device connected to the host

Bits 28:18 Reserved, must be kept at reset value.

Bit 17 **DDISC**: Device connection

– Host mode

This bit is set when a device connection is detected. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 16 **THR512**: 512 byte threshold interrupt

This bit is set to 1 by the hardware when 512 bytes have been transmitted or received during isochronous transfers. This bit is read/write but only 0 can be written and writing 1 has no effect. Note that no information is available to indicate the associated channel/endpoint, however in practice only one ISO endpoint/channel with such large packets can be supported, so that channel.

Bit 15 **CTR**: Completed transfer in host mode

This bit is set by the hardware to indicate that an endpoint/channel has successfully completed a transaction; using DIR and IDN bits software can determine which endpoint/channel requested the interrupt. This bit is read-only.

Bit 14 **PMAOVR:** Packet memory area over / underrun

This bit is set if the microcontroller has not been able to respond in time to an USB memory request. The USB peripheral handles this event in the following way: During reception an ACK handshake packet is not sent, during transmission a bit-stuff error is forced on the transmitted stream; in both cases the host retries the transaction. The PMAOVR interrupt must never occur during normal operations. Since the failed transaction is retried by the host, the application software has the chance to speed-up device operations during this interrupt handling, to be ready for the next transaction retry; however this does not happen during isochronous transfers (no isochronous transaction is anyway retried) leading to a loss of data in this case. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 13 **ERR:** Error

This flag is set whenever one of the errors listed below has occurred:

NANS: No ANSwer. The timeout for a host response has expired.

CRC: Cyclic redundancy check error. One of the received CRCs, either in the token or in the data, was wrong.

BST: Bit stuffing error. A bit stuffing error was detected anywhere in the PID, data, and/or CRC.

FVIO: Framing format violation. A non-standard frame was received (EOP not in the right place, wrong token sequence, etc.).

The USB software can usually ignore errors, since the USB peripheral and the PC host manage retransmission in case of errors in a fully transparent way. This interrupt can be useful during the software development phase, or to monitor the quality of transmission over the USB bus, to flag possible problems to the user (for example loose connector, too noisy environment, broken conductor in the USB cable and so on). This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 12 **WKUP:** Wake-up

This bit is set to 1 by the hardware when, during suspend mode, activity is detected that wakes up the USB peripheral. This event asynchronously clears the SUSPRDY bit in the CTLR register and activates the USB_WAKEUP line, which can be used to notify the rest of the device (for example wake-up unit) about the start of the resume process. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 11 **SUSP:** Suspend mode request

– Device mode

This bit is set by the hardware when no traffic has been received for 3 ms, indicating a suspend mode request from the USB bus. The suspend condition check is enabled immediately after any USB reset and it is disabled by the hardware when the suspend mode is active (SUSPEN=1) until the end of resume sequence. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 10 **RST_DCON:** USB reset request (Device mode) or device connect/disconnect (Host mode)

– Device mode

This bit is set by hardware when an USB reset is released by the host and the bus returns to idle. USB reset state is internally detected after the sampling of 60 consecutive SE0 cycles.

– Host mode

This bit is set by hardware when device connection or device disconnection is detected. Device connection is signaled after J state is sampled for 22 cycles consecutively from unconnected state. Device disconnection is signaled after SE0 state is seen for 22 bit times consecutively from connected state.

Bit 9 **SOF:** Start of frame

This bit signals the beginning of a new USB frame and it is set when a SOF packet arrives through the USB bus. The interrupt service routine can monitor the SOF events to have a 1 ms synchronization event to the USB host and to safely read the USB_FNR register which is updated at the SOF packet reception (this can be useful for isochronous applications). This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 8 **ESOF:** Expected start of frame

– Device mode

This bit is set by the hardware when an SOF packet is expected but not received. The host sends an SOF packet each 1 ms, but if the device does not receive it properly, the suspend timer issues this interrupt. If three consecutive ESOF interrupts are generated (for example three SOF packets are lost) without any traffic occurring in between, a SUSP interrupt is generated. This bit is set even when the missing SOF packets occur while the suspend timer is not yet locked. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 7 **L1REQ:** LPM L1 state request

– Device mode

This bit is set by the hardware when LPM command to enter the L1 state is successfully received and acknowledged. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **DIR:** Direction of transaction

This bit is written by the hardware according to the direction of the successful transaction, which generated the interrupt request.
If DIR bit = 0, VTTX bit is set in the USB_CHEPnR register related to the interrupting endpoint. The interrupting transaction is of IN type (data transmitted by the USB peripheral to the host PC).
If DIR bit = 1, VTRX bit or both VTTX/VTRX are set in the USB_CHEPnR register related to the interrupting endpoint. The interrupting transaction is of OUT type (data received by the USB peripheral from the host PC) or two pending transactions are waiting to be processed. This information can be used by the application software to access the USB_CHEPnR bits related to the triggering transaction since it represents the direction having the interrupt pending. This bit is read-only.

Bits 3:0 **IDN[3:0]:** Device Endpoint / host channel identification number

These bits are written by the hardware according to the host channel or device endpoint number, which generated the interrupt request. If several endpoint/channel transactions are pending, the hardware writes the identification number related to the endpoint/channel having the highest priority defined in the following way: two levels are defined, in order of priority: isochronous and double-buffered bulk channels/endpoints are considered first and then the others are examined. If more than one endpoint/channel from the same set is requesting an interrupt, the IDN bits in USB_ISTR register are assigned according to the lowest requesting register, CHEP0R having the highest priority followed by CHEP1R and so on. The application software can assign a register to each endpoint/channel according to this priority scheme, so as to order the concurring endpoint/channel requests in a suitable way. These bits are read only.

40.6.3 USB frame number register (USB_FNR)

Address offset: 0x48

Reset value: 0x0000 0XXX (where X is undefined)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDP	RXDM	LCK	LSOF[1:0]		FN[10:0]										
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **RXDP:** Receive data + line status

This bit can be used to observe the status of received data plus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event.

Bit 14 **RXDM:** Receive data - line status

This bit can be used to observe the status of received data minus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event.

Bit 13 **LCK:** Locked

– Device mode

This bit is set by the hardware when at least two consecutive SOF packets have been received after the end of an USB reset condition or after the end of an USB resume sequence. Once locked, the frame timer remains in this state until an USB reset or USB suspend event occurs.

Bits 12:11 **LSOF[1:0]:** Lost SOF

– Device mode

These bits are written by the hardware when an ESOF interrupt is generated, counting the number of consecutive SOF packets lost. At the reception of an SOF packet, these bits are cleared.

Bits 10:0 **FN[10:0]:** Frame number

This bit field contains the 11-bits frame number contained in the last received SOF packet. The frame number is incremented for every frame sent by the host and it is useful for isochronous transfers. This bit field is updated on the generation of an SOF interrupt.

40.6.4 USB Device address (USB_DADDR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EF	ADD[6:0]						
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **EF:** Enable function

This bit is set by the software to enable the USB Device. The address of this device is contained in the following ADD[6:0] bits. If this bit is at 0 no transactions are handled, irrespective of the settings of USB_CHEPnR registers.

Bits 6:0 **ADD[6:0]:** Device address

– Device mode

These bits contain the USB function address assigned by the host PC during the enumeration process. Both this field and the endpoint/channel address (EA) field in the associated USB_CHEPnR register must match with the information contained in a USB token in order to handle a transaction to the required endpoint.

– Host mode

These bits contain the address transmitted with the LPM transaction

40.6.5 USB LPM control and status register (USB_LPMCSR)

Address offset: 0x54

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BESL[3:0]				REM WAKE	Res.	LPM ACK	LPM EN
								r	r	r	r	r		rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **BESL[3:0]:** BESL value

– Device mode

These bits contain the BESL value received with last ACKed LPM Token

Bit 3 **REMWAKE:** bRemoteWake value

– Device mode

This bit contains the bRemoteWake value received with last ACKed LPM Token

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LPMACK:** LPM token acknowledge enable

– Device mode:

0: the valid LPM token is NYET.
1: the valid LPM token is ACK.
The NYET/ACK is returned only on a successful LPM transaction:
No errors in both the EXT token and the LPM token (else ERROR)
A valid bLinkState = 0001B (L1) is received (else STALL)

Bit 0 **LPMEN:** LPM support enable

– Device mode

This bit is set by the software to enable the LPM support within the USB Device. If this bit is at 0 no LPM transactions are handled.

40.6.6 USB battery charging detector (USB_BCDR)

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPPU_DPD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2DET	SDET	PDET	Res.	SDEN	PDEN	Res.	BCDEN
rw								r	r	r		rw	rw		rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **DPPU_DPD:** DP pull-up / DPDM pull-down

– Device mode

This bit is set by software to enable the embedded pull-up on DP line. Clearing it to 0 can be used to signal disconnect to the host when needed by the user software.

– Host mode

This bit is set by software to enable the embedded pull-down on DP and DM lines.

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **PS2DET:** DM pull-up detection status

– Device mode

This bit is active only during PD and gives the result of comparison between DM voltage level and V_{LGC} threshold. In normal situation, the DM level must be below this threshold. If it is above, it means that the DM is externally pulled high. This can be caused by connection to a PS2 port (which pulls-up both DP and DM lines) or to some proprietary charger not following the BCD specification.

0: Normal port detected (connected to SDP, ACA, CDP or DCP).

1: PS2 port or proprietary charger detected.

Bit 6 **SDET:** Secondary detection (SD) status

– Device mode

This bit gives the result of SD.

0: CDP detected.

1: DCP detected.

Bit 5 **PDET:** Primary detection (PD) status

– Device mode

This bit gives the result of PD.

0: no BCD support detected (connected to SDP or proprietary device).

1: BCD support detected (connected to ACA, CDP or DCP).

Bit 4 Reserved, must be kept at reset value.

Bit 3 **SDEN:** Secondary detection (SD) mode enable

– Device mode

This bit is set by the software to put the BCD into SD mode. Only one detection mode (PD, SD or OFF) must be selected to work correctly.

Bit 2 **PDEN:** Primary detection (PD) mode enable

– Device mode

This bit is set by the software to put the BCD into PD mode. Only one detection mode (PD, SD or OFF) must be selected to work correctly.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **BCDEN:** Battery charging detector (BCD) enable

– Device mode

This bit is set by the software to enable the BCD support within the USB Device. When enabled, the USB PHY is fully controlled by BCD and cannot be used for normal communication. Once the BCD discovery is finished, the BCD must be placed in OFF mode by clearing this bit to 0 in order to allow the normal USB operation.

40.6.7 USB endpoint/channel n register (USB_CHEPnR)

Address offset: 0x00 + 0x4 * n, (n = 0 to 7)

Reset value: 0x0000 0000

The USB peripheral supports up to 8 bidirectional endpoints or host channels. Each USB Device must support a control endpoint/channel whose address (EA bits) must be set to 0. The USB peripheral behaves in an undefined way if multiple endpoints are enabled having the same endpoint/channel number value. For each endpoint, an USB_CHEPnR register is available to store the endpoint/channel specific information.

They are also reset when an USB reset is received from the USB bus or forced through bit USBRST in the CTLR register, except the VTRX and VTTX bits, which are kept unchanged to avoid missing a correct packet notification immediately followed by an USB reset event. Each endpoint/channel has its USB_CHEPnR register where *n* is the endpoint/channel identifier.

Read-modify-write cycles on these registers must be avoided because between the read and the write operations some bits can be set by the hardware and the next write would modify them before the CPU has the time to detect the change. For this purpose, all bits affected by this problem have an ‘invariant’ value that must be used whenever their modification is not required. It is recommended to modify these registers with a load instruction where all the bits, which can be modified only by the hardware, are written with their ‘invariant’ value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	THREE_ERR_RX[1:0]		THREE_ERR_TX[1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]						
	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	rc_w0	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VTRX	DTOGRX	STATRX[1:0]		SETUP	UTYPE[1:0]		EP KIND	VTTX	DTOGTX	STATTX[1:0]		EA[3:0]			
rc_w0	t	t	t	r	rw	rw	rw	rc_w0	t	t	t	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **THREE_ERR_RX[1:0]:** Three errors for an IN transaction

– Host mode

This bit is set by the hardware when 3 consecutive transaction errors occurred on the USB bus for an IN transaction. THREE_ERR_RX is not generated for isochronous transactions. The software can only clear this bit.

Coding of the received error:

00: Less than 3 errors received.

01: More than 3 errors received, last error is timeout error.

10: More than 3 errors received, last error is data error (CRC error).

11: More than 3 errors received, last error is protocol error (invalid PID, false EOP, bitstuffing error, SYNC error).

Bits 28:27 **THREE_ERR_TX[1:0]:** Three errors for an OUT or SETUP transaction

– Host mode

This bit is set by the hardware when 3 consecutive transaction errors occurred on the USB bus for an OUT transaction. THREE_ERR_TX is not generated for isochronous transactions. The software can only clear this bit.

Coding of the received error:

00: Less than 3 errors received.

01: More than 3 errors received, last error is timeout error.

10: More than 3 errors received, last error is data error (CRC error).

11: More than 3 errors received, last error is protocol error (invalid PID, false EOP, bitstuffing error, SYNC error).

Bit 26 **ERR_RX:** Received error for an IN transaction

– Host mode

This bit is set by the hardware when an error (for example no answer by the device, CRC error, bit stuffing error, framing format violation, etc.) has occurred during an IN transaction on this channel. The software can only clear this bit. If the ERRM bit in USB_CNTR register is set, a generic interrupt condition is generated together with the channel related flag, which is always activated.

Bit 25 **ERR_TX:** Received error for an OUT/SETUP transaction

– Host mode

This bit is set by the hardware when an error (for example no answer by the device, CRC error, bit stuffing error, framing format violation, etc.) has occurred during an OUT or SETUP transaction on this channel. The software can only clear this bit. If the ERRM bit in USB_CNTR register is set, a generic interrupt condition is generated together with the channel related flag, which is always activated.

Bit 24 **LS_EP:** Low speed endpoint – host with HUB only

– Host mode

This bit is set by the software to send an LS transaction to the corresponding endpoint.

0: Full speed endpoint

1: Low speed endpoint

Bit 23 **NAK:**

– Host mode

This bit is set by the hardware when a device responds with a NAK. Software can use this bit to monitor the number of NAKs received from a device.

Bits 22:16 **DEVADDR[6:0]:**

– Host mode

Device address assigned to the endpoint during the enumeration process.

Bit 15 **VTRX:** USB valid transaction received

– Device mode

This bit is set by the hardware when an OUT/SETUP transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated. The type of occurred transaction, OUT or SETUP, can be determined from the SETUP bit described below.

A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.

This bit is read/write but only 0 can be written, writing 1 has no effect.

– Host mode

This bit is set by the hardware when an IN transaction is successfully completed on this channel. The software can only clear this bit. If the CTRM bit in USB_CNTR register is set a generic interrupt condition is generated together with the channel related flag, which is always activated.

- A transaction ended with a NAK sets this bit and NAK answer is reported to application reading the NAK state from the STATRX field of this register. One NAKed transaction keeps pending and is automatically retried by the host at the next frame, or the host can immediately retry by resetting STATRX state to VALID.

- A transaction ended by STALL handshake sets this bit and the STALL answer is reported to application reading the STALL state from the STATRX field of this register. Host application must consequently disable the channel and re-enumerate.

- A transaction ended with ACK handshake sets this bit

If double buffering is disabled, ACK answer is reported by application reading the DISABLE state from the STATRX field of this register. Host application must read received data from USBRAM and re-arm the channel by writing VALID to the STATRX field of this register.

If double buffering is enabled, ACK answer is reported by application reading VALID state from the STATRX field of this register. Host application must read received data from USBRAM and toggle the DTOGTX bit of this register.

- A transaction ended with error sets this bit.

Errors can be seen via the bits ERR_RX (host mode only).

This bit is read/write but only 0 can be written, writing 1 has no effect.

Bit 14 **DTOGRX:** Data Toggle, for reception transfers

If the endpoint/channel is not isochronous, this bit contains the expected value of the data toggle bit (0 = DATA0, 1 = DATA1) for the next data packet to be received. Hardware toggles this bit, when the ACK handshake is sent following a data packet reception having a matching data PID value; if the endpoint is defined as a control one, hardware clears this bit at the reception of a SETUP PID received from host (in device mode), while it sets this bit to 1 when SETUP transaction is acknowledged by device (in host mode).
If the endpoint/channel is using the double-buffering feature this bit is used to support packet buffer swapping too (Refer to *Section 40.5.3: Double-buffered endpoints and usage in Device mode*).
If the endpoint/channel is isochronous, this bit is used only to support packet buffer swapping for data transmission since no data toggling is used for this kind of channels/endpoints and only DATA0 packet are transmitted (Refer to *Section 40.5.5: Isochronous transfers in Device mode*). Hardware toggles this bit just after the end of data packet reception, since no handshake is used for isochronous transfers.
This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force specific data toggle/packet buffer usage. When the application software writes 0, the value of DTOGRX remains unchanged, while writing 1 makes the bit value toggle. This bit is read/write but it can be only toggled by writing 1.

Bits 13:12 **STATRX[1:0]:** Status bits, for reception transfers

– Device mode

These bits contain information about the endpoint status, which are listed in *Table 393: Reception status encoding on page 1567*. These bits can be toggled by software to initialize their value. When the application software writes 0, the value remains unchanged, while writing 1 makes the bit value to toggle. Hardware sets the STATRX bits to NAK when a correct transfer has occurred (VTRX = 1) corresponding to a OUT or SETUP (control only) transaction addressed to this endpoint, so the software has the time to elaborate the received data before it acknowledges a new transaction.

Double-buffered bulk endpoints implement a special transaction flow control, which control the status based upon buffer availability condition (Refer to *Section 40.5.3: Double-buffered endpoints and usage in Device mode*).

If the endpoint is defined as isochronous, its status can be only “VALID” or “DISABLED”, so that the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STATRX bits to ‘STALL’ or ‘NAK’ for an isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing 1.

– Host mode

These bits are the host application controls to start, retry, or abort host transactions driven by the channel.

These bits also contain information about the device answer to the last IN channel transaction and report the current status of the channel according to the following STATRX table of states:

- DISABLE

  DISABLE value is reported in case of ACK acknowledge is received on a single-buffer channel. When in DISABLE state the channel is unused or not active waiting for application to restart it by writing VALID. Application can reset a VALID channel to DISABLE to abort a transaction. In this case the transaction is immediately removed from the host execution list. If the aborted transaction was already under execution it is regularly terminated on the USB but the relative VTRX interrupt is not generated.

- VALID

  A host channel is actively trying to submit USB transaction to device only when in VALID state.VALID state can be set by software or automatically by hardware on a NAKED channel at the start of a new frame. When set to VALID, an host channel enters the host execution queue and waits permission from the host frame scheduler to submit its configured transaction.

  VALID value is also reported in case of ACK acknowledge is received on a double-buffered channel. In this case the channel remains active on the alternate buffer while application needs to read the current buffer and toggle DTOGTX. In case software is late in reading and the alternate buffer is not ready, the host channel is automatically suspended transparently to the application. The suspended double buffered channel is re-activated as soon as delay is recovered and DTOGTX is toggled.

- NAK

  NAK value is reported in case of NAK acknowledge received. When in NAK state the channel is suspended and does not try to transmit. NAK state is moved to VALID by hardware at the start of the next frame, or software can change it to immediately retry transmission by writing it to VALID, or can disable it and abort the transaction by writing DISABLE

- STALL

  STALL value is reported in case of STALL acknowledge received. When in STALL state the channel behaves as disabled. Application must not retry transmission but reset the USB and re-enumerate.

Bit 11 **SETUP:** Setup transaction completed

– Device mode

This bit is read-only and it is set by the hardware when the last completed transaction is a SETUP. This bit changes its value only for control endpoints. It must be examined, in the case of a successful receive transaction (VTRX event), to determine the type of transaction occurred. To protect the interrupt service routine from the changes in SETUP bits due to next incoming tokens, this bit is kept frozen while VTRX bit is at 1; its state changes when VTRX is at 0. This bit is read-only.

– Host mode

This bit is set by the software to send a SETUP transaction on a control endpoint. This bit changes its value only for control endpoints. It is cleared by hardware when the SETUP transaction is acknowledged and VTTX interrupt generated.

Bits 10:9 **UTYPE[1:0]:** USB type of transaction

These bits configure the behavior of this endpoint/channel as described in *Table 394: Endpoint/channel type encoding*. Channel0/Endpoint0 must always be a control endpoint/channel and each USB function must have at least one control endpoint/channel which has address 0, but there can be other control channels/endpoints if required. Only control channels/endpoints handle SETUP transactions, which are ignored by endpoints of other kinds. SETUP transactions cannot be answered with NAK or STALL. If a control endpoint/channel is defined as NAK, the USB peripheral does not answer, simulating a receive error, in the receive direction when a SETUP transaction is received. If the control endpoint/channel is defined as STALL in the receive direction, then the SETUP packet is accepted anyway, transferring data and issuing the CTR interrupt. The reception of OUT transactions is handled in the normal way, even if the endpoint/channel is a control one. Bulk and interrupt endpoints have very similar behavior and they differ only in the special feature available using the EPKIND configuration bit.
The usage of isochronous channels/endpoints is explained in *Section 40.5.5: Isochronous transfers in Device mode*

Bit 8 **EPKIND:** endpoint/channel kind

The meaning of this bit depends on the endpoint/channel type configured by the UTYPE bits. *Table 395* summarizes the different meanings.
DBL_BUF: This bit is set by the software to enable the double-buffering feature for this bulk endpoint. The usage of double-buffered bulk endpoints is explained in *Section 40.5.3: Double-buffered endpoints and usage in Device mode*.
STATUS_OUT: This bit is set by the software to indicate that a status out transaction is expected: in this case all OUT transactions containing more than zero data bytes are answered ‘STALL’ instead of ‘ACK’. This bit can be used to improve the robustness of the application to protocol errors during control transfers and its usage is intended for control endpoints only. When STATUS_OUT is reset, OUT transactions can have any number of bytes, as required.

Bit 7 **VTTX:** Valid USB transaction transmitted

– Device mode

This bit is set by the hardware when an IN transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in the USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated.
A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.
This bit is read/write but only 0 can be written.

– Host mode

Same as VTRX behavior but for USB OUT and SETUP transactions.

Bit 6 **DTOGTX:** Data toggle, for transmission transfers

If the endpoint/channel is non-isochronous, this bit contains the required value of the data toggle bit (0 = DATA0, 1 = DATA1) for the next data packet to be transmitted. Hardware toggles this bit when the ACK handshake is received from the USB host, following a data packet transmission. If the endpoint/channel is defined as a control one, hardware sets this bit to 1 at the reception of a SETUP PID addressed to this endpoint (in device mode) or when a SETUP transaction is acknowledged by the device (in host mode).
If the endpoint/channel is using the double buffer feature, this bit is used to support packet buffer swapping too (Refer to *Section 40.5.3: Double-buffered endpoints and usage in Device mode*).
If the endpoint/channel is isochronous, this bit is used to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (refer to *Section 40.5.5: Isochronous transfers in Device mode*). Hardware toggles this bit just after the end of data packet transmission, since no handshake is used for isochronous transfers.
This bit can also be toggled by the software to initialize its value (mandatory when the endpoint/channel is not a control one) or to force a specific data toggle/packet buffer usage. When the application software writes 0, the value of DTOGTX remains unchanged, while writing 1 makes the bit value to toggle. This bit is read/write but it can only be toggled by writing 1.

Bits 5:4 **STATTX[1:0]:** Status bits, for transmission transfers

– Device mode

These bits contain the information about the endpoint status, listed in *Table 396*. These bits can be toggled by the software to initialize their value. When the application software writes 0, the value remains unchanged, while writing 1 makes the bit value to toggle. Hardware sets the STATTX bits to NAK, when a correct transfer has occurred (VTTX = 1) corresponding to a IN or SETUP (control only) transaction addressed to this channel/endpoint. It then waits for the software to prepare the next set of data to be transmitted.
Double-buffered bulk endpoints implement a special transaction flow control, which controls the status based on buffer availability condition (Refer to *Section 40.5.3: Double-buffered endpoints and usage in Device mode*).
If the endpoint is defined as isochronous, its status can only be “VALID” or “DISABLED”. Therefore, the hardware cannot change the status of the channel/endpoint/channel after a successful transaction. If the software sets the STATTX bits to ‘STALL’ or ‘NAK’ for an isochronous channel/endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing 1.

– Host mode

The STATTX bits contain the information about the channel status. Refer to *Table 396* for the full descriptions (“Host mode” descriptions). Whereas in Device mode, these bits contain the status that are given out on the following transaction, in Host mode they capture the status last received from the device. If a NAK is received, STATTX contains the value indicating NAK.

Bits 3:0 **EA[3:0]:** endpoint/channel address

– Device mode

Software must write in this field the 4-bit address used to identify the transactions directed to this endpoint. A value must be written before enabling the corresponding endpoint.

– Host mode

Software must write in this field the 4-bit address used to identify the channel addressed by the host transaction.

Table 393. Reception status encoding

STATRX[1:0]	Meaning
00	**DISABLED:** all reception requests addressed to this endpoint/channel are ignored.
01	**STALL**: Device mode: the endpoint is stalled and all reception requests result in a STALL handshake. Host mode: this indicates that the device has STALLed the channel.
10	**NAK**: Device mode: the endpoint is NAKed and all reception requests result in a NAK handshake. Host mode: this indicates that the device has NAKed the reception request.
11	**VALID**: this endpoint/channel is enabled for reception.

Table 394. Endpoint/channel type encoding

UTYPE[1:0]	Meaning
00	BULK
01	CONTROL
10	ISO
11	INTERRUPT

Table 395. Endpoint/channel kind meaning

UTYPE[1:0]		EPKIND meaning
00	BULK	DBL_BUF
01	CONTROL	STATUS_OUT
10	ISO	SBUF_ISO: This bit is set by the software to enable the single-buffering feature for isochronous endpoint
11	INTERRUPT	Not used

Table 396. Transmission status encoding

STATTX[1:0]	Meaning
00	**DISABLED:** all transmission requests addressed to this endpoint/channel are ignored.
01	**STALL**: Device mode: the endpoint is stalled and all transmission requests result in a STALL handshake. Host mode: this indicates that the device has STALLed the channel.

Table 396. Transmission status encoding (continued)

STATTX[1:0]	Meaning
10	**NAK**: Device mode: the endpoint is NAKed and all transmission requests result in a NAK handshake. Host mode: this indicates that the device has NAKed the transmission request.
11	**VALID**: this endpoint/channel is enabled for transmission.

40.6.8 USB register map

The table below provides the USB register map and reset values.

Table 397. USB register map and reset values

Offset	Register name	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
0x00	USB_CHEP0R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	USB_CHEP1R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	USB_CHEP2R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USB_CHEP3R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USB_CHEP4R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USB_CHEP5R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 397. USB register map and reset values (continued)

Offset	Register name	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
0x18	USB_CHEP6R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	USB_CHEP7R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20-0x3F	Reserved																																
0x40	USB_CNTR	HOST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISCM	THR512M	CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RST_DCONM	SOFM	ESOFM	L1REQM	Res.	L1RES	L2RES	SUSPEN	SUSPRDY	PDWN	USBRST
	Reset value	0														0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	1	1
0x44	USB_ISTR	Res.	LS_DCON	DCON_STAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISC	THR512	CTR	PMAOVR	ERR	WKUP	SUSP	RST_DCON	SOF	ESOF	L1REQ	Res.	Res.	DIR	IDN[3:0]			
	Reset value		0	0												0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x48	USB_FNR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDP	RXDM	LCK	LSOF [1:0]		FN[10:0]										
	Reset value																	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x
0x4C	USB_DADDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EF	ADD[6:0]						
	Reset value																									0	0	0	0	0	0	0	0
0x54	USB_LPMCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BESL[3:0]				REMWAKE	Res.	LPMACK	LPMEN
	Reset value																									0	0	0	0	0		0	0
0x58	USB_BCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DPPU_DPD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2DET	SDET	PDET	Res.	SDEN	PDEN	Res.	BCDEN
	Reset value																	0								0	0	0		0	0		0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

40.7 USBSRAM registers

Note: The buffer descriptor table is located inside the packet buffer memory in the separate "USBSRAM" address space.

Although the buffer descriptor table is located inside the packet buffer memory ("USBSRAM" area), its entries can be considered as additional registers used to configure the location and size of the packet buffers used to exchange data between the USB macro cell and the device.

The first packet memory location is located at USBSRAM base address. The buffer descriptor table entry associated with the USB_CHEPnR registers is described below. The memory must be addressed using Word (32-bit) accesses.

A thorough explanation of packet buffers and the buffer descriptor table usage can be found in *Structure and usage of packet buffers on page 1534*.

40.7.1 Channel/endpoint transmit buffer descriptor n (USBSRAM_CHEP_TXRXBD_n)

Address offset: 0x0 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register does not program the use of double buffering working in receive mode (otherwise refer to following register description)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TX[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **COUNT_TX[9:0]:** Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it.

Bits 15:0 **ADDR_TX[15:0]:** Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

40.7.2 Channel/endpoint receive buffer descriptor n [alternate] (USBSRAM_CHEP_TXRXBD_n)

Address offset: 0x0 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register programs the use of double buffering and activates receive buffers (otherwise refer to previous register description).

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint/channel descriptor and it is normally defined during

the enumeration process according to its maxPacketSize parameter value (see “Universal Serial Bus Specification”).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_RX[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BLSIZE:** Block size

This bit selects the size of memory block used to define the allocated buffer area.

– If BLSIZE = 0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.

– If BLSIZE = 1, the memory block is 32-byte large, which permits to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

Bits 30:26 **NUM_BLOCK[4:0]:** Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BLSIZE value as illustrated in *Table 398*.

Bits 25:16 **COUNT_RX[9:0]:** Reception byte count

These bits contain the number of bytes received by the endpoint/channel associated with the USB_CHEPnR register during the last OUT/SETUP transaction addressed to it.

Note: Although the application only needs to read this value, it is writable.

Bits 15:0 **ADDR_RX[15:0]:** Reception buffer address

These bits point to the starting address of the packet buffer, which contains the data received by the endpoint/channel associated with the USB_CHEPnR register at the next OUT/SETUP token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

Table 398. Definition of allocated buffer memory

Value of NUM_BLOCK[4:0]	Memory allocated when BLSIZE=0	Memory allocated when BLSIZE=1
0 (00000)	Not allowed	32 bytes
1 (00001)	2 bytes	64 bytes
2 (00010)	4 bytes	96 bytes
3 (00011)	6 bytes	128 bytes
...	...	...
14 (01110)	28 bytes	480 bytes
15 (01111)	30 bytes	512
16 (10000)	32 bytes	544
...	...	...

Table 398. Definition of allocated buffer memory (continued)

Value of NUM_BLOCK[4:0]	Memory allocated when BLSIZE=0	Memory allocated when BLSIZE=1
29 (11101)	58 bytes	...
30 (11110)	60 bytes	992 bytes
31 (11111)	62 bytes	1023 bytes

40.7.3 Channel/endpoint receive buffer descriptor n (USBSRAM_CHEP_RXTXBD_n)

Address offset: 0x4 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register does not program use of double buffering in the transmit mode (otherwise refer to following register description).

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint/channel descriptor and it is normally defined during the enumeration process according to its maxPacketSize parameter value (see “Universal Serial Bus Specification”).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_RX[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BLSIZE:** Block size

This bit selects the size of memory block used to define the allocated buffer area.

– If BLSIZE = 0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.

– If BLSIZE = 1, the memory block is 32-byte large, which permits to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

Bits 30:26 **NUM_BLOCK[4:0]:** Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BLSIZE value as illustrated in *Table 398*.

Bits 25:16 **COUNT_RX[9:0]:** Reception byte count

These bits contain the number of bytes received by the endpoint/channel associated with the USB_CHEPnR register during the last OUT/SETUP transaction addressed to it.

Note: Although the application only needs to read this value, it is writable.

Bits 15:0 **ADDR_RX[15:0]:** Reception buffer address

These bits point to the starting address of the packet buffer, which contains the data received by the endpoint/channel associated with the USB_CHEPnR register at the next OUT/SETUP token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

40.7.4 Channel/endpoint transmit buffer descriptor n [alternate] (USBSRAM_CHEP_RXTXBD_n)

Address offset: 0x4 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register programs use of double buffering and activates transmit buffers (otherwise refer to previous register description).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TX[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **COUNT_TX[9:0]:** Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it.

Bits 15:0 **ADDR_TX[15:0]:** Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

40.7.5 USBSRAM register map

The table below provides the USB register map and reset values.

Table 399. USBSRAM register map and reset values

Offset	Register name	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
0x00	USBSRAM_CHEP_TXRXBD_0	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x00	USBSRAM_CHEP_TXRXBD_0 [alternate]	BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x04	USBSRAM_CHEP_RXTXBD_0	BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x04	USBSRAM_CHEP_RXTXBD_0 [alternate]	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x08	USBSRAM_CHEP_TXRXBD_1	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x08	USBSRAM_CHEP_TXRXBD_1 [alternate]	BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0C	USBSRAM_CHEP_RXTXBD_1	BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0C	USBSRAM_CHEP_RXTXBD_1 [alternate]	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
. . . .	. . . .	. . . .																															
0x38	USBSRAM_CHEP_TXRXBD_7	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x38	USBSRAM_CHEP_TXRXBD_7 [alternate]	BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x3C	USBSRAM_CHEP_RXTXBD_7	BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x3C	USBSRAM_CHEP_RXTXBD_7 [alternate]	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

41 Debug support (DBG)

41.1 DBG introduction and main features

A comprehensive set of debug features is provided to support software development and system integration:

- Breakpoint debugging of the CPU core in the system
- Code execution tracing
- Software instrumentation

The debug features can be controlled via a JTAG/serial-wire debug access port, using industry-standard debugging tools. A trace port allows data to be captured for logging and analysis.

The following debug features are based on Arm® CoreSight™ components.

- General features:
  - SWJ-DP: JTAG/serial-wire debug port
  - AP access ports
  - ROM tables
  - System control space (SCS)
  - Breakpoint unit (BPU)
  - Data watchpoint and trace unit (DWT)
  - Instrumentation trace macrocell (ITM)
  - Embedded trace macrocell (ETM)
  - Trace port interface unit (TPIU)
  - Cross-trigger interface (CTI)

The debug features are accessible by the debugger via the debug AP access ports.

Additional information can be found in the Arm® documents referenced in *Section 41.12*.

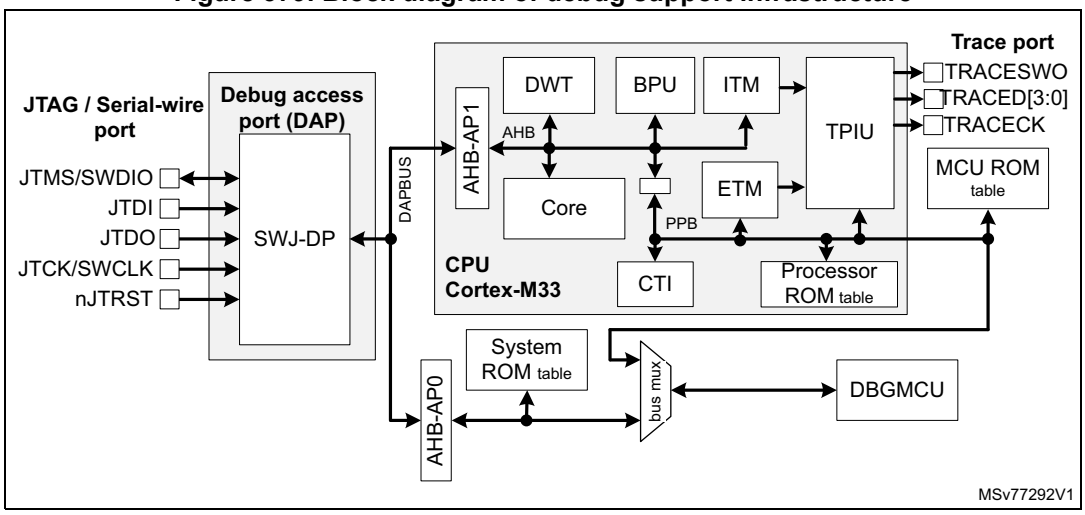
The following debug features are based on STMicroelectronics components

- System debug features:
  - Debug MCU controller (DBGMCU).

41.2 DBG functional description

41.2.1 DBG block diagram

Figure 379. Block diagram of debug support infrastructure



41.2.2 DBG pins and internal signals

Table 400. JTAG/serial-wire debug port pins

Pin name	JTAG debug port		Serial-wire debug port		Pin assignment
	Type	Description	Type	Description	
JTMS/SWDIO	I	JTAG test mode select	I/O	Serial wire data in/out	PA13
JTCK/SWCLK	I	JTAG test clock	I	Serial wire clock	PA14
JTDI	I	JTAG test data input	-	-	PA15
JTDO/TRACESWO(1)	O	JTAG test data output	-	-	PB3
nJTRST	I	JTAG test reset	-	-	PB4

1. Debug access port JTDO and trace port TRACESWO are multiplexed on a single device GPIO pin.

Table 401. Single-wire trace port pins

Pin name	Type	Description	Pin assignment
TRACESWO	O	Single wire trace asynchronous data out	PB3(1)

1. TRACESWO is multiplexed with JTDO. This means that single-wire trace is only available when using the serial-wire debug interface, and not when using JTAG.

Table 402. Embedded trace port pins

Pin name	Type	Description	Pin assignment
TRACED0	O	Embedded trace synchronous data out 0	Refer to the datasheet
TRACED1		Embedded trace synchronous data out 1	
TRACED2		Embedded trace synchronous data out 2	
TRACED3		Embedded trace synchronous data out 3	
TRACECK		Embedded trace synchronous clock out	

41.2.3 DBG reset and clocks

The debug port (SWJ-DP) is reset by a power-on reset and when waking up from Standby mode.

The debugger supplies the clock for the debug port via the debug interface pin JTCK/SWCLK. This clock is used to register the serial input data in both serial-wire and JTAG modes, as well as to operate the state machines and internal logic of the debug port. This clock must therefore continue to toggle for several cycles after the end of an access, to ensure that the debug port returns to the idle state.

The SWJ-DP contains an asynchronous interface to the DAPCLK domain, which covers the rest of the SWJ-DP.

The DAPCLK is a gated version of the CPU bus clock hclk1.

The DAPCLK domain is enabled by the debugger using the CDBGPWRUPREQ bit in the DP_CTRLSTATR register. The clock must be enabled before the debugger can access any of the debug features on the device. The availability of the clock is reflected in the CDBGPWRUPACK bit in the DP_CTRLSTATR register. DAPCLK is disabled at power-up, after OBL, and after a wake-up from Standby. DAPCLK must be disabled when the debugger is disconnected to avoid wasting energy.

The debug components included in the processor are clocked with the processor core clock hclk1.

41.2.4 DBG power domains

The debug components are located in the core power domain. This means that debugger connection is not possible in Standby modes. To avoid losing the connection when the device enters Standby mode, the power to the core can be maintained by setting a bit in the debug unit (DBGMCU). This also keeps the processor clocks active and hold off the reset, so that the debug session is maintained.

41.2.5 DBG low-power modes

The devices include power-saving features that allow the core power domain to be switched off or clocks to be stopped when not required. If the power is switched off, all the debug components are inaccessible to the debugger. When the core is not clocked, all the debug components except the DBGMCU are inaccessible to the debugger. To avoid this, low-power mode emulation is implemented. If emulation is enabled, the device still enters low-power mode, but the clocks and power are maintained. In other words, the device

behaves similarly to when it is in low-power mode, but the debugger does not lose the connection to the CPU.

The low-power emulation mode is programmed in the DBGMCU. For more information refer to *Section 41.13.4: Low-power mode emulation*.

41.2.6 Security

The trace and debug components allow a high degree of access to the processor and system during product development. In order to protect user code and ensure that the debug features cannot be used to alter or compromise the normal operation of the finished product, these features can be disabled or limited in scope. For example, secure software debug and trace can be disabled without preventing the debug of nonsecure code.

The following authentication signals are used by the system to determine which debug features are enabled or disabled:

- **dbgen**: global enable for all debug features

  0: All debug features are disabled.

  1: Debug features in nonsecure state are enabled. Debug features in secure state are dependent on the state of the **spiden** signal.
- **spiden**: enables debug in secure state when **dbgen** = 1.

  0: Debug features are disabled in secure state.

  1: Debug features are enabled in secure state.
- **niden**: enables trace and performance monitoring (noninvasive debug).

  0: Trace generation is disabled.

  1: Trace generation in nonsecure state is enabled. Trace generation in secure state is dependent on the state of the **spniden** signal.
- **spniden**: enables trace and performance monitoring in secure state when **niden** = 1.

  0: Trace generation is disabled in secure state.

  1: Trace generation is enabled in secure state.

For detailed information on the behavior of each component according to the state of the authentication signals, refer to the relevant component section or to the relevant Arm technical documentation.

The state of the signals is set according to the readout protection (RDP) level (see *Section 7.6.2: Readout protection (RDP)*), as shown in the table below:

Table 403. Authentication signal states

RDP level	dbgen	spiden	niden	spniden	Description
0	1	1	1	1	Debug and trace is enabled whatever the state of the processor. The debugger access to secure memory is permitted.
0.5	1	0	1	0	Debug and trace is enabled when the processor is in nonsecure state. The debugger access to secure memory is disabled.
1	1	0	1	0	Debug and trace is enabled when the processor is in nonsecure state. The debugger access to secure memory is disabled, as well as to the following areas: flash memory, SRAM2, backup registers, ICACHE.
2	0	0	0	0	Debug and trace is disabled.

Note: Security features are only relevant when the option bit TZEN = 1. If security features are disabled, the authentication signals are still set according to the RDP level, but since the processor and all memories are nonsecure, spniden and spiden are redundant.

The state of the authentication signals can be read from DAUTHSTATUS register in the system control space (SCS) of the Cortex-M33.

The debugger access to secure memory (when permitted) must be performed using secure transactions on the debug AHB, that is, with PROT[6] set in AP_CSWR.

The debugger access is disabled while the processor is booting from system flash memory (RSS), whatever the RDP level, if security features are enabled (TZEN = 1).

41.2.7 Serial-wire and JTAG debug port

The serial-wire and JTAG debug port (SWJ-DP) is a CoreSight component that implements an external access port for connecting debugging equipment.

The two following types of interface can be configured:

- A 5-pin standard JTAG interface (JTAG-DP)
- A 2-pin (clock + data) serial-wire debug port (SW-DP)

The two modes are mutually exclusive since they share the same I/O pins.

By default the JTAG-DP is selected after a system or a power-on reset. The five I/O pins are configured by hardware in debug alternative function mode.

The SWJ-DP incorporates pull-up resistors on JTDI, JTMS/SWDIO and nJTRST, as well as a pull-down resistor on JTCK/SWCLK.

A debugger can select the SW-DP by transmitting the following serial data sequence on JTMS/SWDIO:

... (50 or more ones) ..., 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, ... (50 or more ones) ...

JTCK/SWCLK must be cycled for each data bit.

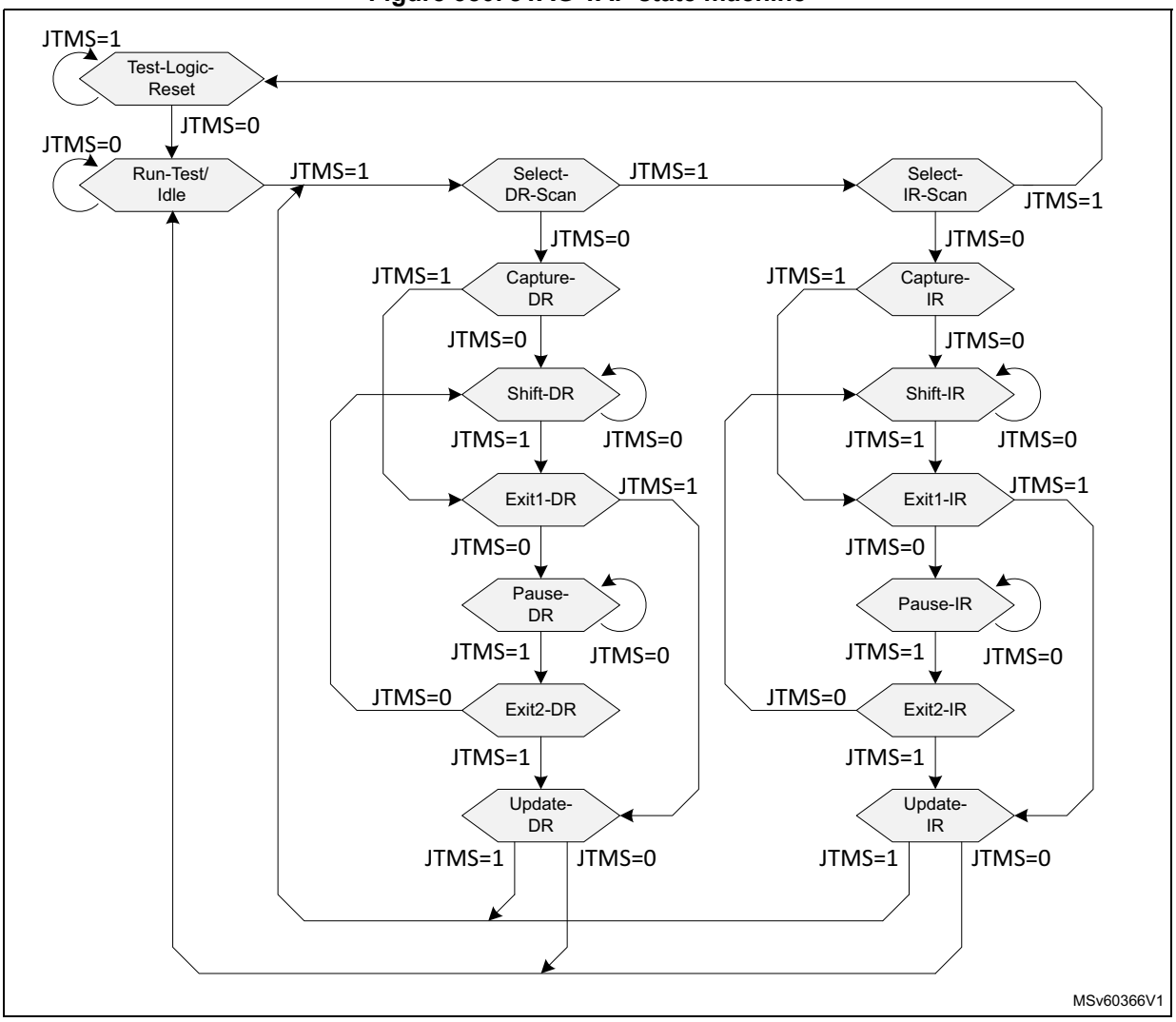
In SW-DP mode, the unused JTAG pins JTDI, JTDO, and nJTRST can be used for other functions.

Note: All SWJ port I/Os can be reconfigured to other functions by software but debugging is no longer possible.

41.2.8 JTAG debug port

The JTAG debug port (JTAG-DP) implements a TAP state machine (TAPSM) based on IEEE Std 1149.1-1990. The state machine controls two scan chains, one associated with an instruction register (IR) and the other one with a number of data registers (DR).

Figure 380. JTAG TAP state machine



The operation of the JTAG-DP is as follows:

1. When the TAPSM goes through the Capture-IR state, 0b0001 is transferred onto the instruction register (IR) scan chain. The IR scan chain is connected between JTDI and JTDO.
2. While the TAPSM is in the Shift-IR state, the IR scan chain shifts one bit for each rising edge of JTCK. This means that, on the first tick:
   – The LSB of the IR scan chain is output on JTDO.
   – Bit[n] of the IR scan chain is transferred to bit[n-1].
   – The value on JTDI is transferred to the MSB of the IR scan chain.
3. When the TAPSM goes through the Update-IR state, the value scanned into the IR scan chain is transferred into the instruction register.
4. When the TAPSM goes through the Capture-DR state, a value is transferred from one of the data registers onto one of the DR scan chains, connected between JTDI and JTDO.
5. The value held in the instruction register determines which data register and associated DR scan chain are selected.
6. This data is then shifted while the TAPSM is in the Shift-DR state, in the same manner as the IR shift in the Shift-IR state.
7. When the TAPSM goes through the Update-DR state, the value scanned into the DR scan chain is transferred into the selected data register.
8. When the TAPSM is in the Run-Test/Idle state, no special actions occur. The IDCODE instruction is loaded in the instruction register.

When active, the nJTRST signal resets the state machine asynchronously to the Test-Logic-Reset state.

The data registers corresponding to the 4-bit IR instructions are listed in *Table 404*.

Table 404. JTAG-DP data registers

IR instruction	Data register	Scan chain length	Description
0000 to 0111	(BYPASS)	1	Not implemented: BYPASS selected
1000	ABORT	35	ABORT register – Bits 34:1 = Reserved – Bit 0 = APABORT: write 1 to generate an AP abort.
1001	(BYPASS)	1	Reserved: BYPASS selected
1010	DPACC	35	Debug port access register Initiates the debug port and gives access to a debug port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request Bits 2:1 = A[3:2] = 2-bit address of a debug port register Bit 0 = RnW = Read request (1) or write request (0) – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data read following a read request Bits 2:0 = ACK[2:0] = 3-bit acknowledge 010 = OK/FAULT 001 = WAIT Others = reserved

Table 404. JTAG-DP data registers (continued)

IR instruction	Data register	Scan chain length	Description
1011	APACC	35	Access port access register Initiates an access port and gives access to an access port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request Bits 2:1 = A[3:2] = 2-bit sub-address of an access port register Bit 0 = RnW= Read request (1) or write request (0) – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data read following a read request Bits 2:0 = ACK[2:0] = 3-bit acknowledge 010 = OK/FAULT 001 = WAIT Others = reserved
1100	(BYPASS)	1	Reserved: BYPASS selected
1101	(BYPASS)	1	Reserved: BYPASS selected
1110	IDCODE	32	ID code 0x0BA0 4477: Cortex-M33 JTAG debug port ID code
1111	BYPASS	1	Bypass A single JTCK cycle delay is inserted between JTDI and JTDO.

Data registers are described in more detail in the Arm® Debug Interface Architecture Specification *[1]*.

41.2.9 Serial-wire debug port

The serial-wire debug (SWD) protocol uses the two following pins:

- SWCLK: clock from host to target
- SWDIO: bidirectional serial data

Serial data is transferred LSB first, synchronously with the clock.

A transfer comprises three phases:

1. Packet request (8-bit) transmitted by the host (see *Table 405*)
2. Acknowledge response (3-bit) transmitted by the target (see *Table 406*)
3. Data transfer (33-bit) transmitted by the host (in case of a write) or target (in case of a read) (see *Table 407*)

The data transfer only occurs if the acknowledge response is OK.

Between each phase, if the direction of the data is reversed, a single clock cycle turn-around time is inserted.

Table 405. Packet request

Bit field	Name	Description
0	Start	Must be 1.
1	APnDP	– 0: DP register access - see *Table 404* for a list of DP registers – 1: AP register access - see *Section 41.4: Access port*
2	RnW	– 0: Write request – 1: Read request
4:3	A(3:2)	Address field of the DP or AP registers
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by host, must be read as 1 by the target.

Table 406. ACK response

Bit field	Name	Description
2:0	ACK	– 000: FAULT – 010: WAIT – 100: OK

Table 407. Data transfer

Bit field	Name	Description
31:0	WDATA or RDATA	Write or read data
32	Parity	Single bit parity of 32 data bits

In the case of a FAULT or WAIT ACK response from the target, the data transfer phase is canceled, unless overrun detection is enabled: in this case the data is ignored by the target (in the case of a write), or not driven (in the case of a read).

A line reset must be generated by the host when it is first connected, or following a protocol error. The line reset consists in 50 or more SWCLK cycles with SWDIO high, followed by two SWCLK cycles with SWDIO low.

For more details on the serial-wire debug protocol, refer to the Arm® Debug Interface Architecture Specification *[1]*.

Note: The SWJ-DP implements SWD protocol version 2.

41.3 Debug port registers

Both SW-DP and JTAG-DP access the debug port (DP) registers listed in *Table 408*.

The debugger accesses the DP registers as follows:

1. If using JTAG, program the A[3:2] field in the DPACC register with the register address within the bank. Program the RnW bit to select a read or write. In the case of a write, program the DATA field with the write data. If using SWD, the A[3:2] and RnW fields are

part of the packet request word sent to the SW-DP with the APnDP bit reset (see *Table 405*). The write data are sent in the data phase.

2. To access one of the banked DP registers at address 0x4, the register number must first be written to the DP_SELECTR register at address 0x8. Any subsequent read or write to address 0x4 accesses the register corresponding to the content of the DP_SELECTR register.

41.3.1 DP identification register [alternate] (DP_DPIDR)

Address offset: 0x00

Reset value: 0x0BE0 2477 (SW-DP), 0x0BE0 1477 (JTAG-DP)

Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN
r	r	r	r	r	r	r	r	r	r	r	r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VERSION[3:0]				DESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **REVISION[3:0]:** Revision code
0x0 (JTAG-DP)
0x0 (SW-DP)

Bits 27:20 **PARTNO[7:0]:** Part number for the debug port
0xBE

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **MIN:** Minimal debug port (MINDP) implementation
0x1 MINDP not implemented (transaction counter and pushed operations are supported)

Bits 15:12 **VERSION[3:0]:** DP architecture version
0x1 (JTAG-DP)
0x2 (SW-DP)

Bits 11:1 **DESIGNER[10:0]:** JEDEC designer identity code
0x23B Arm® JEDEC code

Bit 0 Reserved, must be kept at reset value.

41.3.2 DP abort register [alternate] (DP_ABORTR)

Address offset: 0x00

Reset value: 0x0000 0000

Write only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUNERRCLR	WDERRCLR	STKERRCLR	Res.	DAPABORT
											w	w	w		w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **ORUNERRCLR:** Overrrun error clear
0: No effect
1: Clear STICKYORUN bit in DP_CTRLSTATR

Bit 3 **WDERRCLR:** Write data error clear
0: No effect
1: Clear WDATAERR bit in DP_CTRLSTATR

Bit 2 **STKERRCLR:** Sticky error clear
0: No effect
1: Clear STICKYERR bit in DP_CTRLSTATR

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DAPABORT:** Data AP abort
Aborts current AP transaction if an excessive number of WAIT responses are returned, indicating that the transaction is stalled.
0: No effect
1: Aborts the transaction.

41.3.3 DP control and status register (DP_CTRLSTATR)

Address offset: 0x04

This register is accessible when DP_SELECTR.DPBANKSEL = 0x0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CDBGPWRUPACK	CDBGPWRUPREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		r	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATAERR	READOK	STICKYERR	Res.	Res.	Res.	STICKYORUN	ORUNDETECT
								r	r	rw				rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **CDBGPWRUPACK:** See description in *Section 41.2.6*
0 = DAPCLK gated
1 = DAPCLK enabled

Bit 28 **CDBGPWRUPREQ:** Control of DAPCLK enable request signal
0 = Requests DAPCLK gating
1 = Requests DAPCLK enabled

Bits 27:8 Reserved, must be kept at reset value.

Bit 7 **WDATAERR:** Write data error (read only) in SW-DP
There is a parity or framing error on the data phase of a write, or a write that has been accepted by the DP is then discarded without being submitted to the AP.
This bit is reset by writing 1 to the DP_ABORTR.WDERRCLR bit.
0: No error
1: An error occurred.
This bit is reserved in JTAG-DP.

Bit 6 **READOK:** AP read response (read only) in SW-DP
Indicates the response to the last AP read access.
0: Read not OK
1: Read OK
This bit is reserved in JTAG-DP.

Bit 5 **STICKYERR:** Transaction error (read only in SW-DP, R/W in JTAG-DP)
Indicates that an error occurred in an AP transaction.
0: No error
1: An error occurred.
In SW-DP, STICKYERR bit is read only, reset by writing 1 to the DP_ABORTR.STKERRCLR bit.
In JTAG-DP, STICKYERR bit is read, cleared by writing a 1 to it.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **STICKYORUN:** Overrun (read only in SW-DP, R/W in JTAG-DP)
Indicates that an overrun occurred (new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set.
0: No overrun
1: An overrun occurred.
In SW-DP, STICKYORUN bit is read only, reset by writing 1 to the DP_ABORTR.ORUNERRCLR bit.
In JTAG-DP, STICKYORUN bit is read, cleared by writing a 1 to it.

Bit 0 **ORUNDETECT:** Overrun detection mode enable
0: Overrun detection disabled
1: Overrun detection enabled
In the event of an overrun, the STICKYORUN bit is set and subsequent transactions are blocked until the STICKYORUN bit is cleared.

41.3.4 DP data link control register (DP_DLCR)

Address offset: 0x04

This register is accessible when DP_SELECTR.DPBANKSEL = 0x1

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND[1:0]		WIREMODE[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
						r	r	r	r						

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **TURNROUND[1:0]:** Tristate period for SWDIO
- 0x0: 1 data bit period
- 0x1: 2 data bit periods
- 0x2: 3 data bit periods
- 0x3: 4 data bit periods

Bits 7:6 **WIREMODE[1:0]:** SW-DP SWDIO
- 0x0: Synchronous
- Others: Reserved

Bits 5:0 Reserved, must be kept at reset value.

41.3.5 DP target identification register (DP_TARGETIDR)

Address offset: 0x04

This register is accessible when DP_SELECTR.DPBANKSEL = 0x2

Reset value: 0x14B2 0041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TREVISION[3:0]				TPARTNO[15:4]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPARTNO[3:0]				TDESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **TREVISION[3:0]:** Target revision
- 0x1: Revision A

Bits 27:12 **TPARTNO[15:0]:** Target part number
- 0x4B20: STM32WBA2xxx

Bits 11:1 **TDESIGNER[10:0]:** Target designer JEDEC code.
- 0x020: STMicroelectronics

Bit 0 Reserved, must be kept at reset value.

41.3.6 DP data link protocol identification register (DP_DLPIDR)

Address offset: 0x04

This register is accessible when DP_SELECTR.DPBANKSEL = 0x3

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TINSTANCE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN[3:0]			
												r	r	r	r

Bits 31:28 **TINSTANCE[3:0]:** Target instance number
Defines the instance number for this device in a multi-drop system.
0x0: Instance number 0

Bits 27:4 Reserved, must be kept at reset value.

Bits 3:0 **PROTSVN[3:0]:** Serial-wire debug protocol version
0x1: Version 2

41.3.7 DP resend register (DP_EVENSTATR)

Address offset: 0x04

This register is accessible when DP_SELECTR.DPBANKSEL = 0x4

Reset value: 0x0000 0001

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

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EA:** Event status flag
0: Cortex-M33 processor halted
1: Cortex-M33 processor not halted.

41.3.8 DP resend register (DP_RESENDR)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bits 31:0 **RESEND[31:0]:** Value returned by the last AP read or DP_RDBUFFR read.
Used in the event of a corrupted read transfer.

41.3.9 DP access port select register (DP_SELECTR)

Address offset: 0x08

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w	w												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
								w	w	w	w	w	w	w	w

Bits 31:28 **APSEL[3:0]:** Access port selection
Selects the access port for the next transaction.
0x0: AP0 - System debug access port
0x1: AP1 - Cortex-M33 debug access port
Others: reserved

Bits 27:8 Reserved, must be kept at reset value.

Bits 7:4 **APBANKSEL[3:0]:** AP register bank selection
Selects the 4-word register bank on the active AP for the next transaction.

Bits 3:0 **DPBANKSEL[3:0]:** DP register bank selection
Selects the register at address 0x4 of the debug port.
0x0: DP_CTRLSTATR
0x1: DP_DLCR
0x2: DP_TARGETIDR
0x3: DP_DLPIDR
0x4: DP_EVENTSTAT
Others: reserved

41.3.10 DP read buffer register (DP_BUFFR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bits 31:0 **RDBUFF[31:0]:** Contains the value returned by the last AP read access.

The value returned by an AP read access can either be obtained using a second read access to the same address, which initiates a new transaction on the corresponding bus, or else it can be read from this register, in which case no new AP transaction occurs.

41.3.11 DP register map and reset values

These registers are not on the CPU memory bus and are only accessed through SW-DP and JTAG-DP debug interface.

The debug port address is 2-bit wide, defined in the JTAG-DP register DPACC or SW-DP packet request A[3:2] field.

Table 408. DP register map and reset values

Offset	Register name	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
0x00	DP_DPIDR			REVISION[3:0]													MIN		VERSION[3:0]															Res.
	Reset value	0	0	0	0	1	0	1	1	1	1	1	0				0	X	X	X	X	0	1	0	0	0	1	1	1	0	1	1	
0x00	DP_ABORTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUNERRCLR	WDERRCLR	STKERRCLR	Res.	DAPABORT
	Reset value																												0	0	0		0
0x04(1)	DP_CTRLSTATR	Res.	Res.	CDBGPWRUPACK	CDBGPWRUPREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATAERR	READOK	STICKYERR	Res.	Res.	Res.	STICKYORUN	ORUNDETECT
	Reset value			0	0																					0	0	0				0	0
0x04(2)	DP_DLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND[1:0]		WIREMODE[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																							0	0	0	0						

Table 408. DP register map and reset values (continued)

Offset	Register name	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
0x04[3]	DP_TARGETIDR	TREVISION[3:0]				TPARTNO[15:0]																TDESIGNER[10:0]											Res.
	Reset value	0	0	0	1	0	1	0	0	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
0x04[4]	DP_DLPIDR	TINSTANCE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN[3:0]			
	Reset value	0	0	0	0																									0	0	0	1
0x04[5]	DP_EVENTSTATR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EA
	Reset value																																0
0x08	DP_RESENDR	RESEND[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	DP_SELECTR	APSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
	Reset value	X	X	X	X																					X	X	X	X	X	X	X	X
0x0C	DP_BUFFR	RDBUFF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. DP_SELECTR.DPBANKSEL = 0x0.
2. DP_SELECTR.DPBANKSEL = 0x1.
3. DP_SELECTR.DPBANKSEL = 0x2.
4. DP_SELECTR.DPBANKSEL = 0x3.
5. DP_SELECTR.DPBANKSEL = 0x4

41.4 Access port

There are two access ports (AP) attached to the DP.

- AP0, system debug access port: enables access to the system debug features integrated at device level.
- AP1, Cortex-M33 processor core access port: enables access to the debug and trace features integrated in the Cortex-M33 processor core.

41.4.1 Access port registers

The access ports are of MEM-AP type: the debug and trace component registers are mapped in the address space of the associated bus.

An AP is seen by the debugger as a set of 32-bit registers organized in banks of four registers each. Some of these registers are used to configure or monitor the AP itself, while others are used to perform a transfer on the bus. The AP registers are listed in *Table 409*.

The address of the AP registers is composed of the following fields:

- Bits [7:4]: content of the APBANKSEL[3:0] field in the DP_SELECTR register
- Bits [3:2]: content of the A(3:2) field of the APACC data register in the JTAG-DP or of the SW-DP packet request, depending on the debug interface used
- Bits [1:0]: always set to 0

The content of the APSEL[3:0] defines which MEM-AP is being accessed.

The debugger can access the AP registers as follows:

1. Program in the APSEL[3:0] field to choose the AP and the APBANKSEL[3:0] field to select the register bank to be accessed.
2. If using JTAG, program the A[3:2] field in the APACC register, with the register address within the bank. Program the RnW bit to select a read or write. In the case of a write, program the DATA field with the write data. If using SWD, the A[3:2] and RnW fields are part of the packet request word sent to the SW-DP with the APnDP bit set (see *Table 405*). The write data is sent in the data phase.

The debugger can access the memory-mapped debug component registers through the MEM-AP registers (using the above AP register access procedure) as follows:

1. Program the transaction target address in the AP_TAR register.
2. If necessary, program the AP_CSWR register with the transfer parameters (AddrInc for example).
3. Write to or read from the AP_DRWR register to initiate a bus transaction at the address held in the AP_TAR register. Alternatively, a read or write to the AP_BDxR register triggers an access to address AP_TAR[31:4] + x (allowing up to four consecutive addresses to be accessed without changing the address in the AP_TAR register).

For more detailed information on the MEM-AP, refer to the Arm® Debug Interface Architecture Specification *[1]*.

41.4.2 AP control/status word register (APx_CSWR) (x = 0 to 1)

Address offset: 0x00

Reset value: 0x0100 00X0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PROT6	Res.	Res.	PROT[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw			rw	rw	rw	r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGSTATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
									r	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **PROT6:** secure transfer protection bit [6]
This field sets protection attribute HPROT[6] of the AP bus transfer.
0: Secure transfer
1: Nonsecure transfer

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:24 **PROT[3:0]:** Bus transfer protection
This field sets protection attribute HPROT[3:0] of the AP bus transfer.
0bXXX1: Data transfer
0bXX0X: Unprivileged transfer
0bXX1X: Privileged transfer
0bX0XX: Non-bufferable
0bX1XX: Bufferable
0b0XXX: Non-sharable, no look-up non-modifiable
0b1XXX: Sharable, look-up, modifiable

Bits 23:7 Reserved, must be kept at reset value.

Bit 6 **DBGSTATUS:** Device enable (DEVICEEN) status
Indicates whether the Cortex-M33 AP bus can be accessed by the debugger. This bit reflects the state of dbgen.
0: AP AHB bus transfers blocked.
1: AP AHB bus transfers granted.

Bits 5:4 **ADDRINC[1:0]:** Auto-increment mode
Defines whether AP_TAR address is automatically incremented after a transaction.
0x0: No auto-increment
0x1: Address is incremented by the size in bytes of the transaction (SIZE field).
Others: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]:** Size of next memory access transaction
0x0: Byte (8-bit)
0x1: Halfword (16-bit)
0x2: Word (32-bit)
Others: Reserved

41.4.3 AP transfer address register (APx_TAR) (x = 0 to 1)

Address offset: 0x04

Reset value: 0xXXXX XXXX

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

Bits 31:0 **TA[31:0]:** Address of current transfer

41.4.4 AP data read/write register (APx_DRWR) (x = 0 to 1)

Address offset: 0x0C

Reset value: 0xXXXX XXXX

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

Bits 31:0 **TD[31:0]:** Data of current transfer

41.4.5 AP banked data registers y (APx_BDyR) (x = 0 to 1)

Address offset: 0x10 + 0x4 * y, (y = 0 to 3)

Reset value: 0x0000 0000

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

Bits 31:0 **TBD[31:0]:** Banked data of current transfer to address in AP_TAR
AP_TAR + AP_BDnR address [3:2] + 0b00
The auto address incrementing is not performed on AP_BD[3:0]R.
Banked transfers are only supported for word transfers.

41.4.6 AP configuration register (APx_CFGR) (x = 0 to 1)

Address offset: 0xF4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LD	LA	BE
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LD:** Large data
0: Data not longer than 32-bits supported.

Bit 1 **LA:** Long address
0: Physical addresses not longer than 32-bits supported.

Bit 0 **BE:** Big-endian
0: Only little-endian supported.

41.4.7 AP base address register (APx_BASER) (x = 0 to 1)

Address offset: 0xF8

Reset value: Block 0: 0xE008 0003

Reset value: Block 1: 0xE00F E003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASEADDR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASEADDR[15:12]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
r	r	r	r											r	r

Bits 31:12 **BASEADDR[31:12]:** Base address (bits 31 to 12) of ROM table for the AP
The 12 LSBs are zero since the ROM table must be aligned on a 4-Kbyte boundary.
AP0: 0xE0080 (system ROM table)
AP1: 0xE00FE (MCU ROM table)

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **FORMAT:** Base address register format
1: Arm® debug interface v5

Bit 0 **ENTRYPRESENT:** Debug components presence
Indicates that debug components are present on the access port bus.
1: Debug components are present

41.4.8 AP identification register (APx_IDR) (x = 0 to 1)

Address offset: 0xFC

Reset value: 0x1477 0015

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							CLASS [3]
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLASS[2:0]			Res.	Res.	Res.	Res.	Res.	VARIANT[3:0]				TYPE[3:0]			
r	r	r						r	r	r	r	r	r	r	r

Bits 31:28 **REVISION[3:0]:** Revision
0x1: r0p1

Bits 27:24 **JEDECBANK[3:0]:** JEDEC bank
0x4: Arm®

Bits 23:17 **JEDECCODE[6:0]**: JEDEC code
0x3B: Arm®

Bits 16:13 **CLASS[3:0]:** Memory access port
0x8: MEM_AP Standard register map

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:4 **VARIANT[3:0]:** AP variant
0x1: AHB-AP

Bits 3:0 **TYPE[3:0]:** AP type
0x5: AHB5

41.4.9 Access port register map and reset values

These registers are not on the CPU memory bus and are only accessed through SW-DP and JTAG-DP debug interface.

The access port address is 8-bit wide, defined by DP_SELECTR.APBANKSEL[3:0] field and by JTAG-DP register DPACC or SW-DP packet request A[3:2] field.

Table 409. AP register map and reset values

Offset	Register name	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
0x00	**AP1_CSWR**	Res.	SPROT[6]	Res.	Res.	PROT[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGSTATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
	Reset value		0			0	0	0	1																		X	0	0		0	0	0
0x04	**APx_TAR**	TA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	Reserved	Reserved																															
0x0C	**APx_DRWR**	TD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**APx_BD0R**	TBD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**APx_BD1R**	TBD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**APx_BD2R**	TBD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**APx_BD3R**	TBD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20 to 0xF0	Reserved	Reserved																															
0xF4	**APx_CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LD	LA	BE
	Reset value																														0	0	0
0xF8	**AP0_BASER**	BASEADDR[31:12]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT	
	Reset value	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0										1	1	
0xF8	**AP1_BASER**	BASEADDR[31:12]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT	
	Reset value	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0										1	1	
0xFC	**APx_IDR**	REViISION [3:0]				JEDECBANK [3:0]				JEDECCODE[6:0]							CLASS[3:0]				Res.	Res.	Res.	Res.	Res.	VARIANT[3:0]				TYPE[3:0]			
	Reset value	0	0	0	1	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0						0	0	0	1	0	1	0	1

41.5 System debug AP0 features

The system debug subsystem integrates the following features accessible through the AP0:

- System ROM table
- Debug MCU controller (DBGMCU)

41.5.1 System debug ROM table

The system debug ROM table is a CoreSight component that contains the base addresses of all the debug components accessible via the AP0. This table allows a debugger to discover the topology of the debug system automatically.

There is one ROM table in the system debug subsystem.

1. The system debug ROM table is pointed to by the AP0_BASER register. It contains the base-address pointer for the DBGMCU.

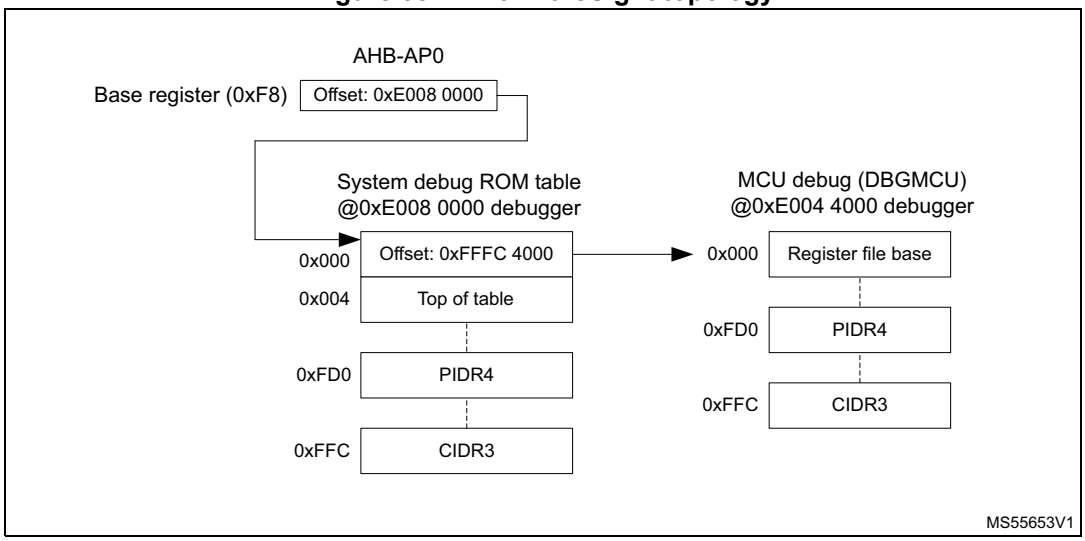
The system debug ROM table (see *Table 410*) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE008 0000 to 0xE008 0FFC.

Table 410. System debug ROM table

Address offset in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0x000	DBGMCU	0xE004 4000	0xFFFC 4000	4	0xFFFC 4003
0x004	Top of table	-	-	-	0x0000 0000
0x00C to 0xFC8	Reserved	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	See *Table 413*

The CoreSight topology for the debug components in the system debug subsystem is shown in *Figure 381*.

Figure 381. AP0: CoreSight topology



41.5.2 System debug memory type register (SYSROM_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0000

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

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM:** system memory
0: No system memory present on this bus

41.5.3 Sytem debug CoreSight peripheral identity register 4 (SYSROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **F4KCOUNT[3:0]:** Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]:** JEP106 continuation code
0x0: STMicroelectronics JEDEC continuation code

41.5.4 System debug CoreSight peripheral identity register 0 (SYSROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00B2

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]:** Part number bits [7:0]
0xB2: STM32WBA2xxx

41.5.5 System debug CoreSight peripheral identity register 1 (SYSROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]:** JEP106 identity code bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x4: STM32WBA2xxx

41.5.6 System debug CoreSight peripheral identity register 2 (SYSROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]:** Component revision number
0x0: Rev r0p0

Bit 3 **JEDEC:** JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]:** JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

41.5.7 System debug CoreSight peripheral identity register 3 (SYSROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]:** Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]:** Customer modified
0x0: No customer modifications

41.5.8 System debug CoreSight component identity register 0 (SYSROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]:** Component ID bits [7:0]
0x0D: Common ID value

41.5.9 System debug CoreSight peripheral identity register 1 (SYSROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** Component ID bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]:** Component ID bits [11:8]
0x0: Common ID value

41.5.10 System debug CoreSight component identity register 2 (SYSROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]:** Component ID bits [23:16]
0x05: Common ID value

41.5.11 System debug CoreSight component identity register 3 (SYSROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]:** Component ID bits [31:24]
0xB1: Common ID value

41.5.12 System debug ROM table register map and reset values

Offset	Register name	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
0xFCC	**ROM_MEMTYPER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																0
0xFD0	**SYSROM_PIDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4-0xFDC	Reserved	Reserved.																															
0xFE0	**SYSROM_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									1	0	0	1	0	0	1	0
0xFE4	**SYSROM_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE8	**SYSROM_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	0
0xFEC	**SYSROM_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	**SYSROM_CIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	**SYSROM_CIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	**SYSROM_CIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	**SYSROM_CIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 41.5: System debug AP0 features* for the register boundary addresses.

41.6 Cortex-M33 AP1 features

The Cortex-M33 subsystem integrates the following features accessible through the AP1:

- MCU and processor ROM tables
- System control space (SCS)
- Breakpoint unit (FPB)
- Data watchpoint and trace unit (DWT)
- Instrumental trace macrocell (ITM)
- Embedded trace macrocell (ETM)
- Trace port interface unit (TPIU)
- Cross-trigger interface (CTI)

41.6.1 CPU ROM tables

The ROM tables are CoreSight components that contain the base addresses of all the debug components accessible via the AP1. These tables allow the debugger to discover the CoreSight topology of the system automatically.

There are two ROM tables on AP1 in the Cortex-M33 subsystem.

1. The MCU ROM table is pointed to by the AP_BASER register in the AP1. It contains the base-address pointers for the processor ROM table and for the TPIU and DBGMCU.
2. The processor ROM table contains the base-address pointer for the system control space (SCS) registers that allows the debugger to identify the CPU core, as well as for the BPU, ITM, and CTI.

The MCU ROM table occupies a 4-Kbyte, 32-bit wide chunk of address space, at base address 0xE00F E000.

Table 411. MCU ROM table

Address offset in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0x000	Processor ROM table	0xE00F F000	0x0000 1000	4	0x0000 1003
0x004	TPIU	0xE004 0000	0xFFF4 2000	4	0xFFF4 2003
0x008	DBGMCU	0xE004 4000	0xFFF4 6000	4	0xFFF4 6003
0x00C	Reserved	-	-	-	0x1FF0 2002
0x010	Top of table	-	-	-	0x0000 0000
0x014 to 0xFC8	Reserved	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	See *Table 413*

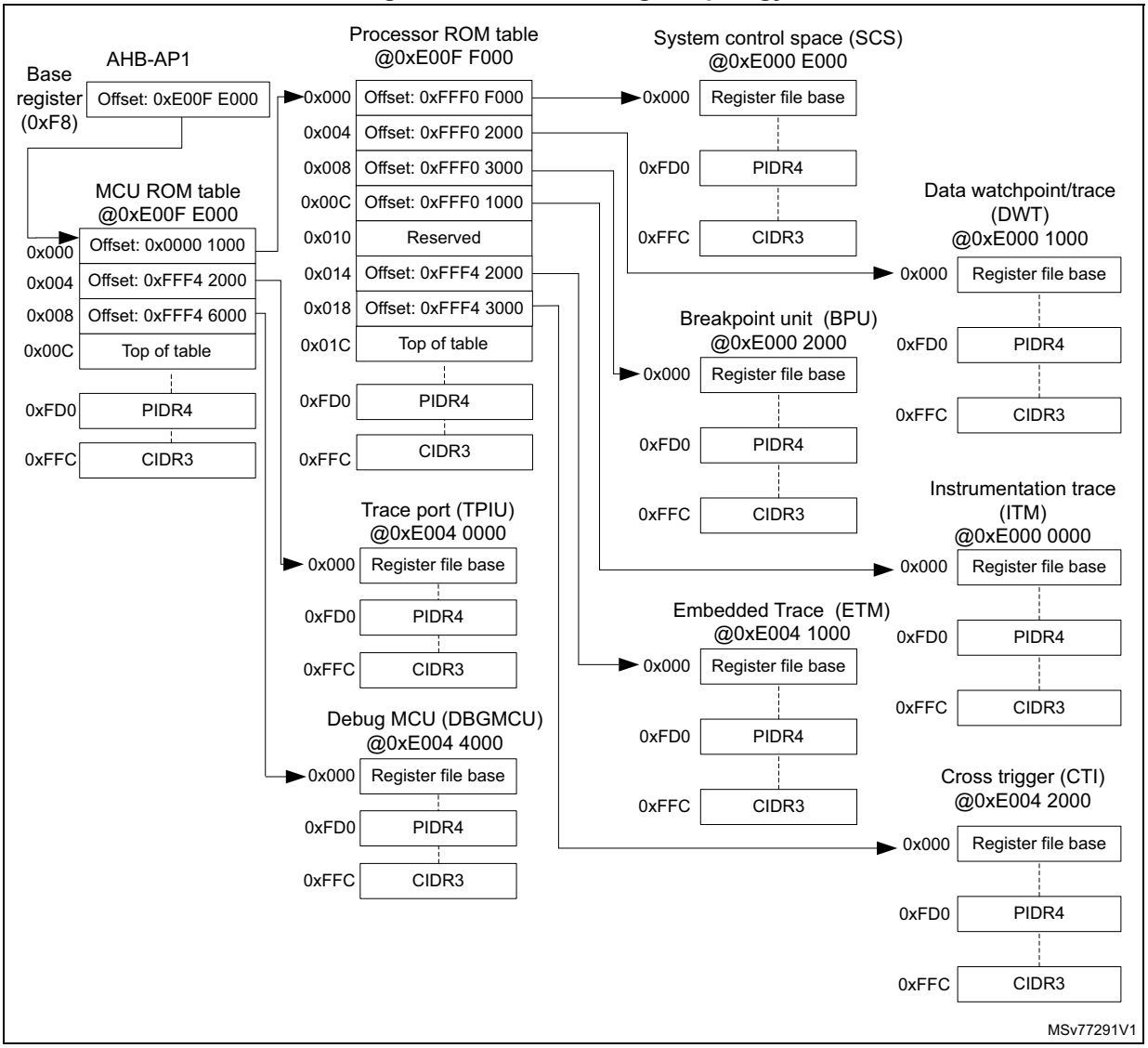
The processor ROM table occupies a 4-Kbyte, 32-bit wide chunk of address space, at base address 0xE00F F000.

Table 412. Processor ROM table

Address in ROM table	Component name	Component base address	Component address offset	Size	Entry
0x000	SCS	0xE000 E000	0xFFF0 F000	4 KB	0xFFF0 F003
0x004	DWT	0xE000 1000	0xFFF0 2000	4 KB	0xFFF0 2003
0x008	BPU	0xE000 2000	0xFFF0 3000	4 KB	0xFFF0 3003
0x00C	ITM	0xE000 0000	0xFFF0 1000	4 KB	0xFFF0 1003
0x010	Reserved	-	-	-	0xFFF4 1002
0x014	ETM	0xE004 1000	0xFFF4 2000	4	0xFFF4 2003
0x018	CTI	0xE004 2000	0xFFF4 3000	4	0xFFF4 3003
0x01C	Top of table	-	-	-	0x0000 0000
0x020 to 0xFC8	Reserved	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	See *Table 413*

The CoreSight topology for the debug components in the CPU subsystem is shown in *Figure 382*.

Figure 382. CPU CoreSight topology



41.6.2 MCU and processor ROM memory type register (ROM_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0001

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

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM:** System memory
1: System memory present on this bus

41.6.3 MCU and processor ROM CoreSight peripheral identity register 4 (ROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000 (MCU ROM)

Reset value: 0x0000 0004 (processor ROM)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **F4KCOUNT[3:0]:** Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]:** JEP106 continuation code
MCU ROM:
0x0: STMicroelectronics JEDEC continuation code
Processor ROM:
0x4: Arm® JEDEC continuation code

41.6.4 MCU and processor ROM CoreSight peripheral identity register 0 (ROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00B2 (MCU ROM)

Reset value: 0x0000 00C9 (processor ROM)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]:** Part number bits [7:0]
MCU ROM:
0xB2: STM32WBA2
processor ROM:
0xC9: Cortex-M33

41.6.5 MCU and processor ROM CoreSight peripheral identity register 1 (ROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0004 (MCU ROM)

Reset value: 0x0000 00B4 (processor ROM)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]:** JEP106 identity code bits [3:0]
0x0: MCU ROM: STMicroelectronics JEDEC code
0xB: Processor ROM: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x4: MCU ROM: STM32WBA2
0x4: Processor ROM: Cortex-M33

41.6.6 MCU and processor ROM CoreSight peripheral identity register 2 (ROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A (MCU ROM)

Reset value: 0x0000 000B (processor ROM)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]:** Component revision number
0x0: Rev r0p0

Bit 3 **JEDEC:** JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]:** JEP106 identity code bits [6:4]
MCU ROM:
0x2: STMicroelectronics JEDEC code
Processor ROM:
0x3: Arm® JEDEC code

41.6.7 MCU and processor ROM CoreSight peripheral identity register 3 (ROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]:** Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]:** Customer modified
0x0: No customer modifications

41.6.8 MCU and processor ROM CoreSight component identity register 0 (ROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]:** Component ID bits [7:0]
0x0D: Common ID value

41.6.9 MCU and processor ROM CoreSight peripheral identity register 1 (ROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** Component ID bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]:** Component ID bits [11:8]
0x0: Common ID value

41.6.10 MCU and processor ROM CoreSight component identity register 2 (ROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]:** Component ID bits [23:16]
0x05: Common ID value

41.6.11 MCU and processor ROM CoreSight component identity register 3 (ROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]:** Component ID bits [31:24]
0xB1: Common ID value

41.6.12 MCU and processor ROM tables register map and reset values

Table 413. ROM table register map and reset values

Offset	Register name	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
0xFCC	**ROM_MEMTYPER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1
0xFD0	**ROM_PIDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	MCU Reset value																									0	0	0	0	0	0	0	0
	processor Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved																																
0xFE0	**ROM_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	MCU Reset value																									1	0	0	1	0	0	1	0
	processor Reset value																									1	1	0	0	1	0	0	1
0xFE4	**ROM_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	MCU Reset value																									0	0	0	0	0	1	0	0
	processor Reset value																									1	0	1	1	0	1	0	0
0xFE8	**ROM_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	MCU Reset value																									0	0	0	0	1	0	1	0
	processor Reset value																									0	0	0	0	1	0	1	1
0xFEC	**ROM_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	**ROM_CIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	**ROM_CIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	**ROM_CIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	**ROM_CIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 41.6: Cortex-M33 AP1 features* for the register boundary addresses.

41.7 Data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as one of the following function:

- Watchpoint
- PC sampling trigger
- Data address sampling trigger
- Data comparator (comparator 1 only)
- Clock cycle counter comparator (comparator 0 only)

It also contains counters for:

- Clock cycles
- Folded instructions
- Load store unit (LSU) operations
- Sleep cycles
- Number of cycles per instruction
- Interrupt overhead

A DWT comparator compares the value held in its DWT_COMPxR registers with one of the following:

- A data address
- An instruction address
- A data value
- The cycle count value, for comparator 0 only.

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:

- One or more DWT data trace packets, containing one or more of:
  - The address of the instruction that caused a data access
  - An address offset, bits[15:0] of the data access address
  - The matched data value
- A watchpoint debug event, on either the PC value or the accessed data address
- A CMPMATCH[N] event that signals the match outside the DWT unit

A watchpoint debug event either generates a DebugMonitor exception or causes the processor to halt execution and enter Debug state.

For more details on how to use the DWT, refer to the Arm®v8-M Architecture Reference Manual *[4]*.

41.7.1 DWT control register (DWT_CTRLR)

Address offset: 0x000

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCOMP[3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	CYCDISS	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSTINIT[3:0]				POSTPRESET[3:0]				CYCCNTENA
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **NUMCOMP[3:0]:** Number of comparators implemented (read only)
0x4: Four comparators

Bit 27 **NOTRCPKT:** Trace sampling and exception tracing support (read only)
0: Supported

Bit 26 **NOEXTTRIG:** External match signal, CMPMATCH support (read only)
0: Supported

Bit 25 **NOCYCCNT:** Cycle counter support (read only)
0: Supported

Bit 24 **NOPRFCNT:** Profiling counter support (read only)
0: Supported

Bit 23 **CYCDISS:** Cycle counter disable secure
Controls whether the cycle counter is disabled in secure mode
0: No effect
1: Disabled incrementing the cycle counter when the processor is in secure mode

Bit 22 **CYCEVTENA:** Enable for POSTCNT underflow event counter packet generation
0: Disabled
1: Enabled

Bit 21 **FOLDEVTENA:** Enable for folded instruction counter overflow event generation
0: Disabled
1: Enabled

Bit 20 **LSUEVTENA:** Enable for LSU counter overflow event generation
0: Disabled
1: Enabled

Bit 19 **SLEEPEVTENA:** Enable for sleep counter overflow event generation
0: Disabled
1: Enabled

Bit 18 **EXCEVTENA:** Enable for exception overhead counter overflow event generation
0: Disabled
1: Enabled

Bit 17 **CPIEVTENA:** Enable for CPI counter overflow event generation
0: Disabled
1: Enabled

Bit 16 **EXCTRCENA:** Enable for exception trace generation
0: Disabled
1: Enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PCSAMPLENA:** Enable for POSTCNT counter used as a timer for periodic PC sample packet generation
0: Disabled
1: Enabled

Bits 11:10 **SYNCTAP[1:0]:** Synchronization packet counter tap
Selects the position of the synchronization packet counter tap on the CYCCNT counter. This determines the synchronization packet rate.
00: Disabled. No synchronization packets
01: Tap at CYCCNT[24]
10: Tap at CYCCNT[26]
11: Tap at CYCCNT[28]

Bit 9 **CYCTAP:** Selects the position of the POSTCNT tap on the CYCCNT counter.
0: Tap at CYCCNT[6]
1: Tap at CYCCNT[10]

Bits 8:5 **POSTINIT[3:0]:** Initial value of the POSTCNT counter
Writes to this field are ignored if POSTCNT counter is enabled (CYCEVTENA or PCSAMPLENA must be reset prior to writing POSTINIT).

Bits 4:1 **POSTPRESET[3:0]:** Reloads value of the POSTCNT counter.

Bit 0 **CYCCNTENA:** Enable for CYCCNT counter
0: Disabled
1: Enabled

41.7.2 DWT cycle count register (DWT_CYCCNTR)

Address offset: 0x004

Reset value: 0x0000 0000

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

Bits 31:0 **CYCCNT[31:0]:** Processor clock cycle counter

41.7.3 DWT CPI count register (DWT_CPICNTR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CPICNT[7:0]:** CPI counter

Counts additional cycles required to execute multi-cycle instructions (except those recorded by DWT_LSUCNTR) and counts any instruction fetch stalls.

41.7.4 DWT exception count register (DWT_EXCCNTR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **EXCCNT[7:0]**: Exception overhead cycle counter

Counts the number of cycles spent in exception processing.

41.7.5 DWT sleep count register (DWT_SLPCNTR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SLEEPCNT[7:0]:** Sleep cycle counter
Counts the number of cycles spent in sleep mode (WFI, WFE, sleep-on-exit).

41.7.6 DWT LSU count register (DWT_LSUCNTR)

Address offset: 0x014

Reset value: 0x0000 0000

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LSUCNT[7:0]:** load store counter
Counts additional cycles required to execute load and store instructions.

41.7.7 DWT fold count register (DWT_FOLDCNTR)

Address offset: 0x018

Reset value: 0x0000 0000

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **FOLDCNT[7:0]:** Folded instruction counter
Increments on each instruction that takes 0 cycles.

41.7.8 DWT program counter sample register (DWT_PCSR)

Address offset: 0x01C

Reset value: 0x0000 0000

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

Bits 31:0 **EIASAMPLE[31:0]:** Executed Instruction Address sample value
Samples the current value of the program counter.

41.7.9 DWT comparator register x (DWT_COMPxR)

Address offset: 0x020 + 0x10 * x, (x = 0 to 3)

Reset value: 0x0000 0000

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

Bits 31:0 **COMP[31:0]:** Reference value for comparison

41.7.10 DWT function register 0(DWT_FUNCTR0)

Address offset: 0x028

Reset value: 0x5800 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: Capability identification
Identifies the capability for match for comparator 0.
0b01011: Cycle Counter, Instruction Address, Data Address and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: Comparator match

Indicates if a comparator match has occurred since the register was last read.

0: No match

1: A match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Data value size

Defines the size of the object being watched for by Data Value and Data Address comparators.

0x0: 1 byte

0x1: 2 bytes

0x2: 4 bytes

0x3: reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: Action on match

0x0: Trigger only

0x1: Generate debug event

0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.

0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: Match type

Controls the type of match generated by comparator 0.

For possible values of this field, refer to *[4]*.

DWT function register 1 (DWT_FUNCTR1)

Address offset: 0x038

Reset value: 0xD000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: Capability identification

Identifies the capability for match for comparator 1.

0b11010: Instruction Address, Instruction Address Limit, Data Address, Data Address Limit, and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: Comparator match

Indicates if a comparator match has occurred since the register was last read.

0: No match

1: Amatch occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Data value size

Defines the size of the object being watched for by Data Value and Data Address comparators.

0x0: 1 byte

0x1: 2 bytes

0x2: 4 bytes

0x3: Reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: Action on match

0x0: Trigger only

0x1: Generate debug event

0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.

0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: Match type

Controls the type of match generated by comparator 1.

For possible values of this field, refer to *[4]*.

DWT function register 2 (DWT_FUNCTR2)

Address offset: 0x048

Reset value: 0x5000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: capability identification
Identifies the capability for MATCH for comparator 2
0b01010: Instruction Address, Data Address, and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: comparator match
Indicates if a comparator match has occurred since the register was last read.
0: No match
1: Amatch occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Data value size:
Defines the size of the object being watched for by Data Value and Data Address comparators.
0x0: 1 byte
0x1: 2 bytes
0x2: 4 bytes
0x3: Reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: Action on match
0x0: trigger only
0x1: Generate debug event
0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: Match type
Controls the type of match generated by comparator 2.
For possible values of this field, refer to *[4]*.

DWT function register 3 (DWT_FUNCTR3)

Address offset: 0x058

Reset value: 0xF000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: capability identification
Identifies the capability for MATCH for comparator 2.
0b11110: Instruction Address, Instruction Address Limit, Data Address, Data Address Limit, Data value, Linked Data Value, and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: comparator match
Indicates if a comparator match has occurred since the register was last read.
0: No match
1: Amatch occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Data value size
Defines the size of the object being watched for by Data Value and Data Address comparators.
0x0: 1 byte
0x1: 2 bytes
0x2: 4 bytes
0x3: Reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: Action on match
0x0: Trigger only
0x1: Generate debug event
0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: Match type
Controls the type of match generated by comparator 2.
For possible values of this field, refer to *[4]*.

41.7.11 DWT device type architecture register (DWT_DEVARCHR)

Address offset: 0xFC8

Reset value: 0x4770 1A02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]:** Architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B Arm®.

Bit 20 **PRESENT:** DWT_DEVARCH register present
0x1: Present.

Bits 19:16 **REVISION[3:0]:** Architecture revision
0x0: DWT architecture v2.0.

Bits 15:12 **ARCHVER[3:0]:** Architecture version
0x1: DWT architecture v2.0.

Bits 11:0 **ARCHPART[11:0]:** Architecture part
0xA02: DWT architecture

41.7.12 DWT device type register 4 (DWT_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]:** Sub-type
0x0: Other

Bits 3:0 **MAJOR[3:0]:** Major type
0x0: Miscellaneous

41.7.13 DWT CoreSight peripheral identity register 4 (DWT_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **F4KCOUNT[3:0]:** Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]:** JEP106 continuation code
0x4: Arm® JEDEC code

41.7.14 DWT CoreSight peripheral identity register 0 (DWT_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]:** Part number bits [7:0]
0x21: DWT part number

41.7.15 DWT CoreSight peripheral identity register 1 (DWT_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]:** JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]:** Part number bits [11:8]
0xD: DWT part number

41.7.16 DWT CoreSight peripheral identity register 2 (DWT_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]:** Component revision number
0x0: r0p0

Bit 3 **JEDEC:** JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]:** JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

41.7.17 DWT CoreSight peripheral identity register 3 (DWT_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]:** Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]:** Customer modified
0x0: No customer modifications

41.7.18 DWT CoreSight component identity register 0 (DWT_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]:** Component ID bits [7:0]
0x0D: Common ID value

41.7.19 DWT CoreSight peripheral identity register 1 (DWT_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** Component ID bits [15:12] - component class
0x9: Debug component

Bits 3:0 **PREAMBLE[11:8]:** Component ID bits [11:8]
0x0: Common ID value

41.7.20 DWT CoreSight component identity register 2 (DWT_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]:** Component ID bits [23:16]
0x05: Common ID value

41.7.21 DWT CoreSight component identity register 3 (DWT_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]:** Component ID bits [31:24]
0xB1: Common ID value

41.7.22 DWT register map and reset values

Table 414. DWT register map and reset values

Offset	Register name	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
0x000	DWT_CTRLR	NUMCOMP [3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	CYCDISS	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA	Res.	Res.	Res.	PCSAMPLENA	SYNCTAP [1:0]		CYCTAP	POSINIT[3:0]				POSTPRESET [3:0]				CYCCNTENA
	Reset value	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	DWT_CYCCNTR	CYCCNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	DWT_CPICNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x00C	DWT_EXCCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x010	DWT_SLPCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x014	DWT_LSUCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x018	DWT_FOLDCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x01C	DWT_PCSR	EIASAMPLE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	DWT_COMP0R	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	Reserved																																
0x028	DWT_FUNCT0R	ID[4:0]							MATCHED													DATAVSIZE [1:0]					ACTION [1:0]		MATCH[3:0]				
	Reset value	0	1	0	1	1			0													0	0				0	0	0	0	0	0	
0x02C	Reserved																																
0x030	DWT_COMP1R	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	Reserved	Reserved.																															
0x038	DWT_FUNCT1R	ID[4:0]							MATCHED													DATAVSIZE [1:0]					ACTION [1:0]		MATCH[3:0]				
	Reset value	1	1	0	1	0			0													0	0				0	0	0	0	0	0	
0x03C	Reserved																																
0x040	DWT_COMP2R	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	Reserved	Reserved.																															

Table 414. DWT register map and reset values (continued)

Offset	Register name	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
0x048	DWT_FUNCT2R	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE [1:0]		Res.	Res.	Res.	Res.	ACTION [1:0]		MATCH[3:0]			
	Reset value	0	1	0	1	0			0													0	0					0	0	0	0	0	0
0x04C	Reserved	Reserved.																															
0x050	DWT_COMP3R	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	Reserved	Reserved.																															
0x058	DWT_FUNCT3R	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE [1:0]		Res.	Res.	Res.	Res.	ACTION [1:0]		MATCH[3:0]			
	Reset value	1	1	1	1	0			0													0	0					0	0	0	0	0	0
0x05C to 0xFC4	Reserved	Reserved.																															
0xFC8	DWT_DEVARCHR	ARCHTITECT[10:0]											PRESEN	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[11:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0
0xFCC	DWT_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:8]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD0	DWT_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFC4	Reserved	Reserved.																															
0xFE0	DWT_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	DWT_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	DWT_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	DWT_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	DWT_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	DWT_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	DWT_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	DWT_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 411: MCU ROM table* for the register boundary addresses.

41.8 Instrumentation trace macrocell (ITM)

The ITM generates trace information as packets. There are three sources that can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The three sources, in decreasing order of priority, are:

1. Software trace

   Software can write directly to any of 32 × 32-bit ITM stimuli registers to generate packets. The permission level for each port can be programmed. When software writes to an enabled stimulus port, the ITM combines the identity of the port, the size of the write access and the data written, into a packet that it writes to a FIFO. The ITM outputs the packets from the FIFO onto the trace bus. Reading a stimulus port register returns the status of the stimulus register (empty or pending) in bit 0.

2. Hardware trace

   The DWT generates trace packets in response to a data trace event, a PC sample, or a performance profiling counter wraparound. The ITM outputs these packets on the trace bus.

3. Local timestamping

   The ITM contains a 21-bit counter clocked by the (predivided) processor clock. The counter value is output in a timestamp packet on the trace bus. The counter is reset to zero every time a timestamp packet is generated. The timestamps thus indicate the time elapsed since the previous timestamp packet.

For more information on the ITM and how to use it, refer to the Arm®v8-M Architecture Reference Manual *[4]*.

41.8.1 ITM registers

41.8.2 ITM stimulus register x (ITM_STIMRx)

Address offset: 0x000 + 0x4 * x, (x = 0 to 31)

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMULUS[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMULUS[15:2]														STIMULUS1	STIMULUS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	rw	rw

Bits 31:2 **STIMULUS[31:2]:** Trace output data
When writing, written data is output on the trace bus as a software event packet.

Bit 1 **STIMULUS1:** Trace output data bit [1]
When writing, written data is output on the trace bus as a software event packet.
When reading: disable flag:
0: Stimulus port and ITM enabled.
1: Stimulus port and ITM disabled.

Bit 0 **STIMULUS0:** Trace output data bit [0]
When writing, written data is output on the trace bus as a software event packet.
When reading: FIFO ready indicator:
0: Stimulus port buffer is full (or port is disabled).
1: Stimulus port can accept new write data.

41.8.3 ITM trace enable register (ITM_TER)

Address offset: 0xE00

Reset value: 0x0000 0000

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

Bits 31:0 **STIMENA[31:0]:** Enable for stimulus port
Each bit x enables the stimulus port associated with the ITM_STIMRx register.
0: Port disabled
1: Port enabled

41.8.4 ITM trace privilege register (ITM_TPR)

Address offset: 0xE40

Reset value: 0x0000 0000

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRIVMASK[3:0]:** Enables unprivileged access to ITM stimulus ports.
Each bit controls eight stimulus ports.
0bXXX0: Unprivileged access permitted on ports 0 to 7
0bXXX1: Only privileged access permitted on ports 0 to 7
0bXX0X: Unprivileged access permitted on ports 8 to 15
0bXX1X: Only privileged access permitted on ports 8 to 15
0bX0XX: Unprivileged access permitted on ports 16 to 23
0bX1XX: Only privileged access permitted on ports 16 to 23
0b0XXX: Unprivileged access permitted on ports 24 to 31
0b1XXX: Only privileged access permitted on ports 24 to 31.

41.8.5 ITM trace control register (ITM_TCR)

Address offset: 0xE80

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]						
								r	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSPRESCALE[1:0]		Res.	Res.	STALLENA	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
						rw	rw			rw	r	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **BUSY:** Indicates whether the ITM is currently processing events.
0: Not busy
1: Busy

Bits 22:16 **TRACEBUSID[6:0]:** identifier for multi-source trace stream formatting
If multi-source trace is in use, the debugger must write a non-zero value to this field.
Note: Different IDs must be used for each trace source in the system.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **TSPRESCALE[1:0]:** local timestamp prescaler
Used with the trace packet reference clock. The possible values are:
0x0: No prescaling
0x1: Divides by 4.
0x2: Divides by 16.
0x3: Divides by 64.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STALLENA:** Stall enable
0: Drop hardware source packets and generate an overflow if the ITM output is stalled.
1: Stall the processor to guarantee delivery of the data trace packets.

Bit 4 **SWOENA:** Enable for asynchronous clocking of the timestamp counter (read only)
0: Timestamp counter uses processor clock

Bit 3 **TXENA:** Enables forwarding of hardware event packets from the DWT unit to the trace port.
0: Disabled
1: Enabled

Bit 2 **SYNCENA:** Enable for packet transmission synchronization
Note: The debugger setting this bit must also configure the DWT_CTRLR register SYNCTAP field in the DWT for the correct synchronization speed.
0: Disabled
1: Enabled

Bit 1 **TSENA:** Enable for local timestamp generation
0: Disabled
1: Enabled

Bit 0 **ITMENA:** ITM enable
0: Disabled
1: Enabled

41.8.6 ITM device type architecture register (ITM_DEVARCHR)

Address offset: 0xFBC

Reset value: 0x4770 1A01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]:** Architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B Arm®.

Bit 20 **PRESENT:** DWT_DEVARCH register present
0x1: Present.

Bits 19:16 **REVISION[3:0]:** Architecture revision
0x0: DWT architecture v2.0.

Bits 15:12 **ARCHVER[3:0]:** Architecture version
0x1: DWT architecture v2.0.

Bits 11:0 **ARCHPART[11:0]:** Architecture part
0xA01: DWT architecture

41.8.7 ITM device type register 4 (ITM_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0043

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]:** Sub-type
0x4: Associated with a bus, stimulus derived from bus activity

Bits 3:0 **MAJOR[3:0]:** Major type
0x3: Trace source

41.8.8 ITM CoreSight peripheral identity register 4 (ITM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **F4KCOUNT[3:0]:** Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]:** JEP106 continuation code
0x4: Arm® JEDEC code

41.8.9 ITM CoreSight peripheral identity register 0 (ITM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]:** Part number bits [7:0]
0x21: ITM part number

41.8.10 ITM CoreSight peripheral identity register 1 (ITM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]:** JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]:** Part number bits [11:8]
0xD: ITM part number

41.8.11 ITM CoreSight peripheral identity register 2 (ITM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]:** Component revision number
0x0: r0p0

Bit 3 **JEDEC:** JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]:** JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

41.8.12 ITM CoreSight peripheral identity register 3 (ITM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]:** Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]:** Customer modified
0x0: No customer modifications

41.8.13 ITM CoreSight component identity register 0 (ITM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]:** Component ID bits [7:0]
0x0D: Common ID value

41.8.14 ITM CoreSight peripheral identity register 1 (ITM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** Component ID bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]:** Component ID bits [11:8]
0x0: Common ID value

41.8.15 ITM CoreSight component identity register 2 (ITM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]:** Component ID bits [23:16]
0x05: Common ID value

41.8.16 ITM CoreSight component identity register 3 (ITM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]:** Component ID bits [31:24]
0xB1: Common ID value

41.8.17 ITM register map and reset values

Table 415. ITM register map and reset values

Offset	Register name	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
0x000 to 0x07C	ITM_STIMR0 to ITM_STIMR31	STIMULUS[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x080 to 0xDFC	Reserved																																
0xE00	ITM_TER	STIMENA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE04 to 0xE3C	Reserved																																
0xE40	ITM_TPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVMASK [3:0]			
	Reset value																													0	0	0	0
0xE44 to 0xE7C	Reserved																																
0xE80	ITM_TCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	TSPRESCALE [1:0]		Res.	Res.	STALLENA	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
	Reset value									0	0	0	0	0	0	0	0							0	0			0	0	0	0	0	0
0xE84 to 0xFB8	Reserved																																
0xFBC	DWT_DEVARCHR	ARCHTITECT[10:0]											PRESEN	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[11:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	1
0xFC0 to 0xFC8	Reserved																																
0xFCC	DWT_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:8]			
	Reset value																									0	1	0	0	0	0	1	1
0xFD0	ITM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved																																
0xFE0	ITM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	ITM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	ITM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	ITM_PIDR3	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	ITM_CIDR0	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1

Table 415. ITM register map and reset values (continued)

Offset	Register name	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
0xFF4	ITM_CIDR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	1	1	0	0	0	0	0
0xFF8	ITM_CIDR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ITM_CIDR3	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 41.5: System debug AP0 features* for the register boundary addresses.

41.9 Breakpoint unit (BPU)

The BPU allows hardware breakpoints to be set. It contains eight comparators that monitor the instruction fetch address. If a match occurs, the instruction comparators can be configured to generate a breakpoint instruction.

For more information on the BPU and how to use it, refer to the Arm®v8-M Architecture Reference Manual *[4]*.

41.9.1 BPU control register (BPU_CTRLR)

Address offset: 0x000

Reset value: 0x1000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NUM_CODE[6:4]			Res.	Res.	Res.	Res.	NUM_CODE[3:0]				Res.	Res.	KEY	ENABLE
	r	r	r					r	r	r	r			rw	rw

Bits 31:28 **REV[3:0]:** Revision number
0x1: BPU version 2

Bits 27:15 Reserved, must be kept at reset value.

Bits 14:12, 7:4 **NUM_CODE[6:0]:** Number of instruction address comparators supported - least significant bits
0x8: 8 instruction comparators supported

Bits 11:8, 3:2 Reserved, must be kept at reset value.

Bit 1 **KEY:** Write protect key
A write to BPU_CTRLR register is ignored if this bit is not set to 1.

Bit 0 **ENABLE**: BPU enable
0: Disabled
1: Enabled

41.9.2 BPU comparator x register (BPU_COMPxR)

Address offset: 0x008 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BPADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BPADDR[15:1]															BE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:1 **BPADDR[31:1]:** Breakpoint address

Bit 0 **BE:** Breakpoint enable
0: disabled
1: enabled

41.9.3 BPU device type architecture register (BPU_DEVARCHR)

Address offset: 0xFBC

Reset value: 0x4770 1A03

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]:** Architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B Arm®.

Bit 20 **PRESENT:** DEVARCH register present
0x1: Present.

Bits 19:16 **REVISION[3:0]:** Architecture revision
0x0: BPU architecture v2.0.

Bits 15:12 **ARCHVER[3:0]:** Architecture version
0x1: BPU architecture v2.0.

Bits 11:0 **ARCHPART[11:0]:** Architecture part
0xA03: BPU architecture

41.9.4 BPU device type register 4 (BPU_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]:** Sub-type
0x0: Other

Bits 3:0 **MAJOR[3:0]:** Major type
0x0: Miscellaneous

41.9.5 BPU CoreSight peripheral identity register 4 (BPU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **F4KCOUNT[3:0]:** register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]:** JEP106 continuation code
0x4: Arm® JEDEC code

41.9.6 BPU CoreSight peripheral identity register 0 (BPU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]:** Part number bits [7:0]
0x21: BPU part number

41.9.7 BPU CoreSight peripheral identity register 1 (BPU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]:** JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]:** Part number bits [11:8]
0xD: BPU part number

41.9.8 BPU CoreSight peripheral identity register 2 (BPU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]:** Component revision number
0x0: r0p0

Bit 3 **JEDEC:** JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]:** JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

41.9.9 BPU CoreSight peripheral identity register 3 (BPU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]:** Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]:** Customer modified
0x0: No customer modifications

41.9.10 BPU CoreSight component identity register 0 (BPU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]:** Component ID bits [7:0]
0x0D: Common ID value

41.9.11 BPU CoreSight peripheral identity register 1 (BPU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** Component ID bits [15:12] - component class
0x9: Debug component

Bits 3:0 **PREAMBLE[11:8]:** Component ID bits [11:8]
0x0: Common ID value

41.9.12 BPU CoreSight component identity register 2 (BPU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]:** Component ID bits [23:16]
0x05: Common ID value

41.9.13 BPU CoreSight component identity register 3 (BPU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]:** Component ID bits [31:24]
0xB1: Common ID value

41.9.14 BPU register map and reset values

Table 416. BPU register map and reset values

Offset	Register name	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
0x000	BPU_CTRLR	NUM_LIT [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUM_CODE [6:4]			Res.	Res.	Res.	Res.	NUM_CODE [3:0]				Res.	Res.	KEY	ENABLE
	Reset value	0	0	0	1														0	0	0					1	0	0	0			0	0
0x004	Reserved																																
0x008 to 0x024	BPU_COMP0R to BPU_COMP7R	BPADDR[31:1]																															BE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C-0xFB8	Reserved																																
0xFBC	DWT_DEVARCHR	ARCHTITECT[10:0]											PRESEN	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[11:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	1
0xFC0 to 0xFC8	Reserved																																
0xFCC	DWT_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:8]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD0	BPU_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved																																
0xFE0	BPU_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	BPU_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	BPU_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	BPU_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	BPU_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	BPU_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	BPU_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	BPU_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 41.5: System debug AP0 features* for the register boundary addresses.

41.10 Embedded trace macrocell (ETM)

The ETM is a CoreSight component closely coupled to the CPU. The ETM generates trace packets that allow the execution of the Cortex-M33 core to be traced. In STM32WBA2, the ETM is configured for instruction trace only. Data accesses are not included in the trace information.

The ETM receives information from the CPU over the processor trace interface, including:

- Number of instructions executed in the same cycle
- Changes in program flow
- Current processor instruction state
- Addresses of memory locations accessed by load and store instructions
- Type, direction, and size of a transfer
- Condition code information
- Exception information
- Wait for interrupt state information

For more information, refer to the Arm CoreSight ETM-M33 Technical Reference Manual *[6]*.

41.10.1 ETM registers

The ETM registers are located at the address range 0xE004 1000 to 0xE004 1FFC.

ETM programming control register (ETM_PRGCTLR)

Address offset: 0x004

Reset value: 0x0000 0000

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

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EN**: Trace unit enable
0: Disabled
1: Enabled

ETM status register (ETM_STATR)

Address offset: 0x00C

Reset value: 0xXXXX XXXX

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PMSTABLE**: Stability status
Indicates that the ETM-M33 registers are stable and can be read.
0: Not stable
1: Stable

Bit 0 **IDLE**: Trace unit status
Indicates that the trace unit is inactive.
0: Not idle
1: Idle

ETM configuration register (ETM_CONFIGR)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RS	Res.	COND[5:0]						CCI	BB	Res.	Res.	Res.
			rw		rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **RS**: Return stack enable
0: Disabled
1: Enabled

Bit 11 Reserved, must be kept at reset value.

Bits 10:5 **COND[5:0]**: Conditional instruction tracing
0x0: Conditional instruction tracing disabled
0x1: Conditional load instructions traced
0x2: Conditional store instructions traced
0x3: Conditional load and store instructions traced
0x7: All conditional instructions traced

Bit 4 **CCI**: Cycle counting in instruction trace
0: Disabled
1: Enabled

Bit 3 **BB**: Branch broadcast mode
0: Disabled
1: Enabled

Bits 2:0 Reserved, must be kept at reset value.

ETM event control 0 register (ETM_EVENTCTL0R)

Address offset: 0x020

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE1	Res.	Res.	Res.	SEL1[3:0]				TYPE0	Res.	Res.	Res.	SEL0[3:0]			
rw				rw	rw	rw	rw	rw				rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **TYPE1**: Resource type for event1
0: Single selected resource
1: Boolean combined resource pair

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:8 **SEL1[3:0]**: Resource number based on TYPE1
Selects the resource number, based on the value of TYPE1.
When TYPE1 = 0, a single resource from 0-15 defined by SEL1[3:0] is selected.
When TYPE1 = 1, a boolean combined resource pair defined by SEL1[2:0] is selected.

Bit 7 **TYPE0**: resource type for event0
0: Single selected resource
1: Boolean combined resource pair

Bits 6:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL0[3:0]**: resource number based on TYPE0
Selects the resource number, based on the value of TYPE0.
When TYPE0 = 0, a single resource from 0-15 defined by SEL0[3:0] is selected.
When TYPE0 = 1, a boolean combined resource pair defined by SEL0[2:0] is selected.

ETM event control 1 register (ETM_EVENTCTL1R)

Address offset: 0x024

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LPOVERRIDE	ATB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTEN[1:0]	
			rw	rw										rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LPOVERRIDE**: Low-power state behavior override
0: Normal low-power state behavior
1: The resources and event trace generation are not affected by entry to a low-power state.

Bit 11 **ATB**: ATB trigger enable
0: Disabled
1: Enabled

Bits 10:2 Reserved, must be kept at reset value.

Bits 1:0 **INSTEN[1:0]**: instruction event generation
Enables generation of an event element in the instruction stream.
0bX0: Event0 does not cause an event element.
0bX1: Event0 causes an event element when it occurs.
0b0X: Event1 does not cause an event element.
0b1X: Event1 causes an event element when it occurs.

ETM stall control register (ETM_STALLCTLR)

Address offset: 0x02C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	INSTPRIORITY	Res.	ISTALL	Res.	Res.	Res.	Res.	LEVEL[3:0]			
					rw		rw					rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **INSTPRIORITY**: Instruction trace priority
Prioritizes instruction trace if instruction trace buffer space is less than LEVEL[3:0].
0: The ETM must not prioritize instruction trace.
1: The ETM can prioritize instruction trace.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **ISTALL**: Processor stalling

Stalls processor based on instruction trace buffer space.

0: The ETM must not stall the processor.

1: The ETM can stall the processor.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **LEVEL[3:0]**: Threshold at which stalling becomes active

This field provides four levels. This level can be varied to optimize the level of invasion caused by stalling, balanced against the risk of a FIFO overflow.

0x0: Zero invasion, but greater risk of FIFO overflow

...

0xF: Maximum invasion but less risk of FIFO overflow

ETM synchronization period register (ETM_SYNCPR)

Address offset: 0x034

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIOD[4:0]				
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PERIOD[4:0]**: Synchronization period

Defines the number of bytes of trace between trace synchronization requests as a total of the number of bytes generated by the instruction stream.

0xA: 1024 bytes

ETM cycle count control register (ETM_CCCTLR)

Address offset: 0x038

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	THRESHOLD[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **THRESHOLD[11:0]**: Instruction trace cycle count threshold

Sets the threshold value for instruction trace cycle counting. The threshold represents the minimum interval between cycle-count trace packets.

ETM trace identification register (ETM_TRACEIDR)

Address offset: 0x040

Reset value: 0xXXXX XXXX

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

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **TRACEID[6:0]**: Trace identification to output onto the trace bus

This field must be programmed with a unique value to differentiate it from other trace sources in the system.

Values 0x00 and 0x70-0x7F are reserved.

ETM ViewInst main control register (ETM_VICTLR)

Address offset: 0x080

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXLEVEL_S[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TRCERR	TRCRESET	SSSTATUS	Res.	EVENT[7:0]							
				rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **EXLEVEL_S[3:0]**: Exception level in secure state

Controls whether instruction tracing is enabled for the corresponding exception level, in secure state.

0bXXX0: Instruction trace not generated in secure state, for exception level 0

0bXXX1: Instruction trace generated in secure state, for exception level 0

0b0XXX: Instruction trace not generated in secure state, for exception level 3

0b1XXX: Instruction trace generated in secure state, for exception level 3

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **TRCERR**: Trace system error exception

0: The system error exception is traced only if the instruction or exception immediately before the system error exception is traced.

1: The system error exception is always traced.

Bit 10 **TRCRESET**: trace reset exception

0: The reset exception is traced only if the instruction or exception immediately before the reset exception is traced.

1: The reset exception is always traced.

Bit 9 **SSSTATUS**: Start/stop logic status
0: stopped
1: started

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **EVENT[7:0]**: Event selector

ETM counter reload value register 0 (ETM_CNTRLDVR0)

Address offset: 0x140

Reset value: 0xXXXX XXXX

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **VALUE[15:0]**: Counter reload value
This value is loaded in to the counter each time the reload event occurs.

ETM identification register 8 (ETM_IDR8)

Address offset: 0x180

Reset value: 0x0000 0000

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

Bits 31:0 **MAXSPEC[31:0]**: Maximum speculation depth
Indicates the maximum speculation depth of the instruction trace stream. This is the maximum number of P0 elements that have not been committed in the trace stream at any one time.
0x0: The maximum trace speculation depth is zero.

ETM identification register 9 (ETM_IDR9)

Address offset: 0x184

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP0KEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP0KEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP0KEY[31:0]**: Number of P0 right-hand keys used
0x0: No P0 right-hand keys used in instruction trace

ETM identification register 10 (ETM_IDR10)

Address offset: 0x188

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP1KEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP1KEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP1KEY[31:0]**: number of P1 right-hand keys used (including normal and special keys)
0x0: No P1 right-hand keys used in instruction trace

ETM identification register 11 (ETM_IDR11)

Address offset: 0x18C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP1SPC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP1SPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP1SPC[31:0]**: Number of special P1 right-hand keys used
0x0: No special P1 right-hand keys used in any configuration

ETM identification register 12 (ETM_IDR12)

Address offset: 0x190

Reset value: 0x0000 0001

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

Bits 31:0 **NUMCONDKEY[31:0]**: Number of conditional instruction right-hand keys used
0x1: One conditional instruction right-hand keys implemented

ETM identification register 13 (ETM_IDR13)

Address offset: 0x194

Reset value: 0x0000 0000

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

Bits 31:0 **NUMCONDSPC[31:0]**: Number of special conditional instruction right-hand keys used
0x0: No special conditional instruction right-hand keys implemented

ETM implementation specific register 0 (ETM_IMSPECR0)

Address offset: 0x1C0

Reset value: 0x0000 0000

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **SUPPORT[3:0]**: Implementation specific extension support
0x0: No implementation specific extensions are supported

ETM identification register 0 (ETM_IDR0)

Address offset: 0x1E0

Reset value: 0x2800 06E1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	COMMOPT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCEXDATA	QSUPP[1]
		r												r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QSUPP[0]	Res.	CONDTYPE[1:0]		NUMEVENT[1:0]		RETSTACK	Res.	TRCCCI	TRCCOND	TRCBB	TRCDATA[1:0]		INSTP0[1:0]		Res.
r		r	r	r	r	r		r	r	r	r	r	r	r	

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **COMMOPT**: Commit field meaning

Indicates the meaning of the commit field in some packets.

1: Commit mode 1

Bits 28:18 Reserved, must be kept at reset value.

Bit 17 **TRCEXDATA**: Trace data transfers for exceptions

Indicates support for the tracing of data transfers for exceptions and exception returns.

0: Not implemented

Bits 16:15 **QSUPP[1:0]**: Q element support

0: Not supported

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **CONDTYPE[1:0]**: Conditional results tracing

Indicates how conditional results are traced.

0: The trace unit indicates only if a conditional instruction passes or fails its condition code check

Bits 11:10 **NUMEVENT[1:0]**: Number of events supported

0x1: Two events

Bit 9 **RETSTACK**: Return stack support

1: Two entry return stacks

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TRCCCI**: Cycle counting support

1: Cycle counting implemented

Bit 6 **TRCCOND**: Conditional instruction support

1: Conditional instruction tracing implemented

Bit 5 **TRCBB**: Branch broadcast support

1: Branch broadcast tracing implemented

Bits 4:3 **TRCDATA[1:0]**: Data tracing support

0x0: Data tracing not supported

Bits 2:1 **INSTP0[1:0]**: Support for tracing of load and store instructions as P0 elements

0x0: Not supported

Bit 0 Reserved, must be kept at reset value.

ETM identification register 1 (ETM_IDR1)

Address offset: 0x1E4

Reset value: 0x4100 F421

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DESIGNER[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TRCARCHMAJ[3:0]				TRCARCHMIN[3:0]				REVISION[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **DESIGNER[7:0]**: Trace unit designer
0x41: Arm

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:8 **TRCARCHMAJ[3:0]**: Major trace unit architecture version number
0x4: ETMv4

Bits 7:4 **TRCARCHMIN[3:0]**: Minor trace unit architecture version number
0x2: Minor revision 2

Bits 3:0 **REVISION[3:0]**: Implementation revision number
0x1: Implementation revision 1

ETM identification register 2 (ETM_IDR2)

Address offset: 0x1E8

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	CCSIZE[3:0]				DVSIZE[4:0]					DASIZE[4:1]			
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DASIZE[0]	VMIDSIZE[4:0]					CIDSIZE[4:0]					IASIZE[4:0]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **CCSIZE[3:0]**: Cycle counter size
0x0: 12 bits

Bits 24:20 **DVSIZE[4:0]**: Data value size
0x0: Data value size not supported

Bits 19:15 **DASIZE[4:0]**: Data address size.
0x0: Data address size not supported

Bits 14:10 **VMIDSIZE[4:0]**: Virtual machine ID size
0x0: Virtual machine ID tracing not implemented

Bits 9:5 **CIDSIZE[4:0]**: Context ID size
0x0: Context ID tracing not implemented

Bits 4:0 **IASIZE[4:0]**: Instruction address size
0x4: Maximum 32-bit address size

ETM identification register 3 (ETM_IDR3)

Address offset: 0x1EC

Reset value: 0x0F09 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NOOVERFLOW	NUMPROC[2:0]			SYSSTALL	STALLCTL	SYNCPR	TRCERR	Res.	Res.	Res.	Res.	EXLEVEL_S[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CCITMIN[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **NOOVERFLOW**: ETM_STALLCTLR.NOOVERFLOW implementation
0: Not implemented

Bits 30:28 **NUMPROC[2:0]**: number of processors available for tracing
0x0: One processor

Bit 27 **SYSSTALL**: System support for stall control of the processor
1: System supports stall control

Bit 26 **STALLCTL**: Stall control support
1: ETM_STALLCTLR implemented

Bit 25 **SYNCPR**: Trace synchronization period support
1: ETM_SYNCPR is read-only for instruction trace only configuration. The trace synchronization period is fixed.

Bit 24 **TRCERR**: ETM_VICTLR.TRCERR implementation
0x1: Implemented

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **EXLEVEL_S[3:0]**: Privilege levels implementation
0x9: Privilege levels thread and handler implemented

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **CCITMIN[11:0]**: Minimum value that can be programmed to TRCCCCTLR.THRESHOLD
Defines the minimum cycle counting threshold.
0x4: minimum of four-instruction trace cycles

ETM identification register 4 (ETM_IDR4)

Address offset: 0x1F0

Reset value: 0x0011 4000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMVMIDC[3:0]				NUMCIDC[3:0]				NUMSSCC[3:0]				NUMRSPAIR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMPC[3:0]				Res.	Res.	Res.	SUPPDAC	NUMDVC[3:0]				NUMACPAIRS[3:0]			
r	r	r	r				r	r	r	r	r	r	r	r	r

Bits 31:28 **NUMVMIDC[3:0]**: Number of virtual machine ID (VMID) comparators
0x0: VMID comparators not implemented

Bits 27:24 **NUMCIDC[3:0]**: Number of context ID comparators
0x0: Context ID comparators not supported

Bits 23:20 **NUMSSCC[3:0]**: Number of single-shot comparator controls
0x1: One single-shot comparator control implemented

Bits 19:16 **NUMRSPAIR[3:0]**: Number of resource selection pairs
0x1: Two resource selection pairs implemented

Bits 15:12 **NUMPC[3:0]**: Number of processor comparator inputs for the DWT
0x4: Four processor comparator inputs implemented

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SUPPDAC**: Data address comparisons
0: Data address comparisons not supported

Bits 7:4 **NUMDVC[3:0]**: Number of data value comparators
0x0: No data value comparators implemented

Bits 3:0 **NUMACPAIRS[3:0]**: Number of address comparator pairs
0x0: No address comparator pairs implemented

ETM identification register 5 (ETM_IDR5)

Address offset: 0x1F4

Reset value: 0x90C7 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REDFUNCNTR	NUMCNTR[2:0]			NUMSEQSTATE[2:0]			Res.	LPOVERRIDE	ATBTRIG	TRACEIDSIZE[5:0]					
r	r	r	r	r	r	r		r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NUMEXTINSEL[2:0]			NUMEXTIN[8:0]								
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **REDFUNCNTR**: Reduced function counter
1: Counter 0 implemented as a reduced function counter

Bits 30:28 **NUMCNTR[2:0]**: Number of counters
0x1: One counter implemented.

Bits 27:25 **NUMSEQSTATE[2:0]**: Number of sequencer states
0x0: No sequencer states implemented.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **LPOVERRIDE**: Low-power state override support
1: Low-power state override support implemented

Bit 22 **ATBTRIG**: ATB trigger support
1: ATB trigger support implemented

Bits 21:16 **TRACEIDSIZE[5:0]**: Number of bits of trace identification
0x7: 7-bit trace identification implemented

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:9 **NUMEXTINSEL[2:0]**: Number of external input selectors
0x0: No external input selectors implemented.

Bits 8:0 **NUMEXTIN[8:0]**: Number of external inputs
0x004: Four external inputs implemented.

ETM resource register 2 (ETM_RSCTLR2)

Address offset: 0x208

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAIRINV	INV	Res.	GROUP[2:0]		
										rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **PAIRINV**: Result of a combined pair of resources inversion
0: Not inverted
1: Inverted

Bit 20 **INV**: Selected resources inversion
0: Not inverted
1: Inverted

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **GROUP[2:0]**: Group of resources selection
0x0: External input selectors (select 0-3)
0x1: Inputs from processor DWT comparators element (select 0-3)
0x2: Counter at zero (select 0)
0x3: Single-shot comparator (select 0)
Others: Reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **SELECT[7:0]**: More resources selection
Selects one or more resources from the group selected in GROUP[2:0].

ETM resource register 3 (ETM_RSCTLR3)

Address offset: 0x20C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INV	Res.	GROUP[2:0]		
											rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **INV**: Selected resources inversion
0: Not inverted
1: Inverted

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **GROUP[2:0]**: Group of resources selection
0x0: External input selectors (select 0-3)
0x1: Inputs from processor DWT comparators element (select 0-3)
0x2: Counter at zero (select 0)
0x3: Single-shot comparator (select 0)
Others: Reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **SELECT[7:0]**: More resources selection
Selects one or more resources from the group selected in GROUP[2:0].

ETM single-shot comparator control register 0 (ETM_SSCCR0)

Address offset: 0x280

Reset value: 0xXXXX XXXX

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

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RST**: Single-shot comparator reset
Enables the single-shot comparator resource to be reset when it occurs, to enable another comparator match to be detected.
1: Reset enabled

Bits 23:0 Reserved, must be kept at reset value.

ETM single-shot comparator status register 0 (ETM_SSCSR0)

Address offset: 0x2A0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC	DV	DA	INST
												r	r	r	r

Bit 31 **STATUS**: Single-shot comparator status
Indicates whether any of the selected comparators have matched.
0: No match occurred
1: At least one match occurred

Bits 30:4 Reserved, must be kept at reset value.

Bit 3 **PC**: Processor comparator input sensitivity
1: Single-shot comparator sensitive to processor comparator inputs

Bit 2 **DV**: Data value comparator support
0: Single-shot data value comparisons not supported

Bit 1 **DA**: Data address comparator support
0: Single-shot data address comparisons not supported

Bit 0 **INST**: Instruction address comparator support
0: Single-shot instruction address comparisons not supported

ETM single-shot processor comparator input control register 0 (ETM_SSPCICR0)

Address offset: 0x2C0

Reset value: 0xXXXX XXXX

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PC[3:0]**: Processor comparator inputs selection for single-shot control
0xXXX0: Processor comparator input 0 not selected
0xXXX1: Processor comparator input 0 selected
0xXX0X: Processor comparator input 1 not selected
0xXX1X: Processor comparator input 1 selected
0xX0XX: Processor comparator input 2 not selected
0xX1XX: Processor comparator input 2 selected
0x0XXX: Processor comparator input 3 not selected
0x1XXX: Processor comparator input 3 selected

ETM power-down control register (ETM_PDCR)

Address offset: 0x310

Reset value: 0x0000 0000

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

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **PU**: Power-up request
0: Power-up not requested
1: Power-up requested

Bits 2:0 Reserved, must be kept at reset value.

ETM power-down status register (ETM_PDSR)

Address offset: 0x314

Reset value: 0x0000 0003

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **STICKYPD**: Sticky power-down state
0: Trace register power has not been removed since the ETM_PDSR was last read.
1: Trace register power has been removed since the ETM_PDSR was last read.

Bit 0 **POWER**: ETM power-up status
1: ETM powered up

ETM claim tag set register (ETM_CLAIMSETR)

Address offset: 0xFA0

Reset value: 0x0000 000F

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Claim tag bits setting

Write:
0000: No effect
xxx1: Sets bit 0.
xx1x: Sets bit 1.
x1xx: Sets bit 2.
1xxx: Sets bit 3.
Read:
0xF: Indicates there are four bits in claim tag.

ETM claim tag clear register (ETM_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Claim tag bits reset

Write:
0000: No effect
xxx1: Clears bit 0.
xx1x: Clears bit 1.
x1xx: Clears bit 2.
1xxx: Clears bit 3.
Read: Returns current value of claim tag.

ETM authentication status register (ETM_AUTHSTATR)

Address offset: 0xFB8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x2: Secure non-invasive debug disabled
0x3: Secure non-invasive debug enabled

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for nonsecure non-invasive debug
0x2: Nonsecure non-invasive debug disabled
0x3: Nonsecure non-invasive debug enabled

Bits 1:0 **NSID[1:0]**: Security level for nonsecure invasive debug
0x0: Not implemented

ETM device type architecture register (ETM_DEVARCHR)

Address offset: 0xFBC

Reset value: 0x4772 4A13

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: Architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm limited.

Bit 20 **PRESENT**: DEVARCH register presence
0x1: Present

Bits 19:16 **REVISION[3:0]**: Architecture revision
0x2: ETM architecture v4.2

Bits 15:12 **ARCHVER[3:0]**: Architecture version
0x4: ETM architecture v4.2

Bits 11:0 **ARCHPART[11:0]**: Architecture part
0xA13: ETM architecture

ETM CoreSight device type register (ETM_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0013

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Device sub-type identifier
0x1: Processor trace

Bits 3:0 **MAJORTYPE[3:0]**: Device main type identifier
0x3: Trace source

ETM CoreSight peripheral identity register 4 (ETM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

ETM CoreSight peripheral identity register 0 (ETM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x21: ETM part number

ETM CoreSight peripheral identity register 1 (ETM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0xD: ETM part number

ETM CoreSight peripheral identity register 2 (ETM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x1: r0p1

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm JEDEC code

ETM CoreSight peripheral identity register 3 (ETM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

ETM CoreSight component identity register 0 (ETM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

ETM CoreSight peripheral identity register 1 (ETM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x9: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

ETM CoreSight component identity register 2 (ETM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

ETM CoreSight component identity register 3 (ETM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

41.10.2 ETM register map and reset values

The ETM registers are accessed by the debugger through AP1 at address range 0xE004 1000 to 0xE004 1FFC.

Table 417. ETM register map and reset values

Offset	Register name	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
0x004	ETM_PRGCTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
	Reset value																																0
0x008	Reserved																																
0x00C	ETM_STATR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PMSTABLE	IDLE
	Reset value																															X	X
0x010	ETM_CONFIGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RS	Res.	COND[5:0]						CCI	BB	Res.	Res.	Res.
	Reset value																				X		X	X	X	X	X	X	X	X			
0x014-0x01C	Reserved																																
0x020	ETM_EVENTCTL0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPE1	Res.	Res.	Res.	SEL1[3:0]				TYPE0	Res.	Res.	Res.	SEL0[3:0]			
	Reset value																	X				X	X	X	X	X				X	X	X	X
0x024	ETM_EVENTCTL1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPOVERRIDE	ATB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTEN[1:0]	
	Reset value																				X	X										X	X
0x028	Reserved																																
0x02C	ETM_STALLCTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTPRIORITY	Res.	ISTALL	Res.	Res.	Res.	Res.	LEVEL[3:0]			
	Reset value																						X		X					X	X	X	X
0x030	Reserved																																
0x034	ETM_SYNCPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIOD[4:0]				
	Reset value																												0	1	0	1	0
0x038	ETM_CCCTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THRESHOLD[11:0]											
	Reset value																					X	X	X	X	X	X	X	X	X	X	X	X
0x03C	Reserved																																
0x040	ETM_TRACEIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[6:0]						
	Reset value																										X	X	X	X	X	X	X
0x044-0x07C	Reserved																																

Table 417. ETM register map and reset values (continued)

Offset	Register name	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
0x080	ETM_VICTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXLEVEL_S [3:0]				Res.	Res.	Res.	Res.	TRCERR	TRCRESET	SSTATUS	Res.	EVENT[7:0]							
	Reset value													X	X	X	X					X	X	X		X	X	X	X	X	X	X	X
0x084-0x13C	Reserved																																
0x140	ETM_CNTRLDVR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VALUE[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x144 to 0x17C	Reserved																																
0x180	ETM_IDR8	MAXSPEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x184	ETM_IDR9	NUMP0KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x188	ETM_IDR10	NUMP1KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18C	ETM_IDR11	NUMP1SPC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x190	ETM_IDR12	NUMCONDKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x194	ETM_IDR13	NUMCONDSPC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x198-0x1BC	Reserved																																
0x1C0	ETM_IMSPECR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUPPORT [3:0]			
	Reset value																													0	0	0	0
0x1C4-0x1DC	Reserved																																
0x1E0	ETM_IDR0	Res.	Res.	COMMOPT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCEXDATA	QSUPP[1:0]		Res.	CONDTYPE [1:0]		NUMEVENT [1:0]		RETSTACK	Res.	TRCCCI	TRCCOND	TRCBB	TRCDATA[1:0]		INSTP0[1:0]		Res.
	Reset value			1												0	0	0		0	0	0	1	1		1	1	1	0	0	0	0	
0x1E4	ETM_IDR1	DESIGNER[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCARCHMAJ [3:0]				TRCARCHMIN [3:0]				REVISION [3:0]			
	Reset value	0	1	0	0	0	0	0	1													0	1	0	0	0	0	1	0	0	0	0	1
0x1E8	ETM_IDR2	Res.	Res.	Res.	CCSIZE[3:0]				DVSIZE[4:0]					DASIZE[4:0]				VMIDSIZE[4:0]				CIDSIZE[4:0]				IASIZE[4:0]							
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0x1EC	ETM_IDR3	NOOVERFLOW	NUMPROC[2:0]		SYSSTALL	STALLCTL	SYNCPR	TRCERR	Res.	Res.	Res.	Res.	EXLEVEL_S [3:0]				Res.	Res.	Res.	Res.	CCITMIN[11:0]													
	Reset value	0	0	0	0	1	1	1	1					1	0	0	1					0	0	0	0	0	0	0	0	0	1	0	0

Table 417. ETM register map and reset values (continued)

Offset	Register name	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
0x1F0	ETM_IDR4	NUMVMIDC [3:0]				NUMCIDC [3:0]				NUMSSCC [3:0]				NUMRSPAIR [3:0]				NUMPC [3:0]				Res.	Res.	Res.	SUPPDAC	NUMDVC [3:0]				NUMACPAIRS [3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0				0	0	0	0	0	0	0	0	0
0x1F4	ETM_IDR5	REDFUNCNTR	NUMCNTR[2:0]			NUMSEQSTATE [2:0]			Res.	LPOVERRIDE	ATBTRIG	TRACEIDSIZE [5:0]						Res.	Res.	Res.	Res.	NUMEXTINSEL [2:0]			NUMEXTIN[8:0]								
	Reset value	1	0	0	1	0	0	0		1	1	0	0	0	1	1	1					0	0	0	0	0	0	0	0	0	1	0	0
0x1F8-0x204	Reserved																																
0x208	ETM_RSCTLR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAIRINV	INV	Res.	GROUP [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
	Reset value											X	X		X	X	X									X	X	X	X	X	X	X	X
0x20C	ETM_RSCTLR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INV	Res.	GROUP [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
	Reset value												X		X	X	X									X	X	X	X	X	X	X	X
0x210-0x27C	Reserved																																
0x280	ETM_SSCCR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								X																								
0x284-0x29C	Reserved																																
0x2A0	ETM_SSCSR0	STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC	DV	DA	INST
	Reset value	X																												X	X	X	X
0x2A4-0x2BC	Reserved																																
0x2C0	ETM_SSPCICR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC[3:0]			
	Reset value																													X	X	X	X
0x2C4-0x30C	Reserved																																
0x310	ETM_PDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU	Res.	Res.	Res.
	Reset value																													0			
0x314	ETM_PDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STICKYPD	POWER
	Reset value																															1	1
0x318-0xF9C	Reserved																																
0xFA0	ETM_CLAIMSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													1	1	1	1
0xFA4	ETM_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3:0]			
	Reset value																													0	0	0	0

Table 417. ETM register map and reset values (continued)

Offset	Register name	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
0xFA8-0xFB4	Reserved																																
0xFB8	ETM_AUTHSTATR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID [1:0]		SID [1:0]		NSNID [1:0]		NSID [1:0]	
	Reset value																									X	X	X	X	X	X	X	X
0xFBC	ETM_DEVARCHR	ARCHITECT[10:0]											PRESEN	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[11:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0	1	0	0	1	0	1	0	0	0	0	1	0	0	1	1
0xFC0-0xFC8	Reserved																																
0xFCC	ETM_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE [3:0]				MAJORTYP [3:0]			
	Reset value																									0	0	0	1	0	0	1	1
0xFD0	ETM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved																																
0xFE0	ETM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	ETM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	ETM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	ETM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	ETM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	ETM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	ETM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ETM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 41.5: System debug AP0 features* for the register boundary addresses.

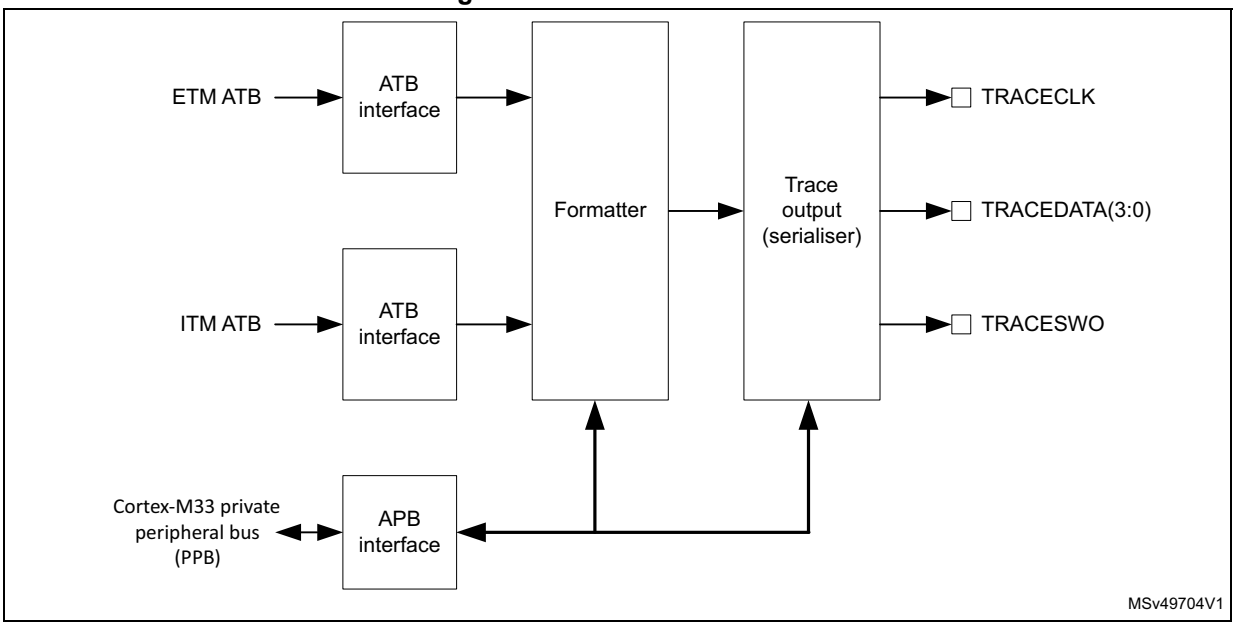
41.11 Trace port interface unit (TPIU)

The TPIU formats the trace stream and outputs it on the external trace port signals. The TPIU has one ATB slave ports for incoming trace data from the ITM and ETM. The trace port is the serial-wire output, TRACESWO and a synchronous parallel port, comprising a clock

output, TRACECLK, and four data outputs, TRACEDATA[3:0]. The trace port width is programmable in the range 1 to 4. Using a smaller port width reduces the number of test points/connector pins needed, and frees up I/Os for other purposes, at the expenses of bandwidth restriction of the trace port, and hence of the quantity of trace information that can be output in real time.

Figure 383 shows the TPIU architecture.

Figure 383. TPIU architecture



For more information on the TPIU and how to use it, refer to the Arm® Cortex-M33 Technical Reference Manual *[5]*.

41.11.1 TPIU registers

41.11.2 TPIU supported port size register (TPIU_SSPSR)

Address offset: 0x000

Reset value: 0x0000 0001

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

Bits 31:0 **PORTSIZE[31:0]:** Supported trace port sizes, from 1 to 32 pins
Bit n-1 when set indicates that port size n is supported.
0x0000 0001: Port size 1 supported

41.11.3 TPIU current port size register (TPIU_CSPSR)

Address offset: 0x004

Reset value: 0x0000 0001

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

Bits 31:0 **PORTSIZE[31:0]:** Current trace port size

Bit n-1 when set indicates that the current port size is n pins. The value of n must be within the range of supported port size (1, 2, and 4). Only one bit can be set, or unpredictable behavior may result. This register must be modified only when the formatter is stopped.

41.11.4 TPIU asynchronous clock prescaler register (TPIU_ACPR)

Address offset: 0x010

Reset value: 0x0000 0000

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

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **PRESCALER[12:0]:** Selects the baud rate for the asynchronous output, TRACESWO

The baud rate is given by the TRACELKIN frequency divided by (PRESCALER +1).

41.11.5 TPIU selected pin protocol register (TPIU_SPPR)

Address offset: 0x0F0

Reset value: 0x0000 0001

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

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **TXMODE[1:0]:** Selects the protocol used for trace output
0x0: Parallel trace port mode
0x1: Asynchronous SWO using Manchester encoding
0x2: Asynchronous SWO using NRZ encoding
0x3: Reserved

41.11.6 TPIU formatter and flush status register (TPIU_FFSR)

Address offset: 0x300

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FTNONSTOP:** Indicates whether formatter can be stopped or not
1: Formatter cannot be stopped.

Bit 2 **TCPRESENT:** Indicates whether the optional TRACECTL output pin is available for use
0: TRACECTL pin is not present in this device.

Bit 1 **FTSTOPPED:** Stop request signal received
The formatter received a stop request signal and all trace data and post-amble is sent. Any additional trace data on the ATB interface is ignored.
0: Formatter not stopped
1: Formatter stopped

Bit 0 **FLINPROG:** Flush in progress
This bit indicates whether a flush on the ATB slave port is in progress and reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the TPIU_FFCR register.
0: No flush in progress
1: Flush in progress

41.11.7 TPIU formatter and flush control register (TPIU_FFCR)

Address offset: 0x304

Reset value: 0x0000 0102

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGIN	Res.	FONMAN	Res.	Res.	Res.	Res.	ENFCONT	Res.
							r		rw					rw	

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **TRIGIN:** Trigger on trigger in
1: Indicates a trigger in the trace stream when the TRIGIN input is asserted.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FONMAN:** Flush on manual
0: Flush completed
1: Generate a flush

Bits 5:2 Reserved, must be kept at reset value.

Bit 1 **ENFCONT:** Continuous formatting enable
Setting this bit to 0 in SWO mode bypasses the formatter and only ITM/DWT trace is output, ETM trace is disabled.
0: Continuous formatting disabled
1: Continuous formatting enabled

Bit 0 Reserved, must be kept at reset value.

41.11.8 TPIU formatter synchronization counter register (TPIU_FSCR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PSCOUNT[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **PSCOUNT[12:0]:** Formatter frames counter
Enables effective use of different sized TPAs without wasting large amounts of the storage capacity of the capture device. This counter contains the number of formatter frames since the last synchronization packet of 128 bits. It is a 12-bit counter with a maximum count value of 4096. This equates to synchronization every 65536 bytes, that is, 4096 packets x 16 bytes per packet. The default is set up for a synchronization packet every 1024 bytes, that is every 64 formatter frames. If the formatter is configured for continuous mode, full and half-word synchronization frames are inserted during normal operation. Under these circumstances, the count value is the maximum number of complete frames between full synchronization packets.

41.11.9 TPIU claim tag set register (TPIU_CLAIMSETR)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]:** Sets claim tag bits

Write:
0000: No effect
xxx1: Sets bit 0.
xx1x: Sets bit 1.
x1xx: Sets bit 2.
1xxx: Sets bit 3.
Read:
0xF: Indicates there are four bits in claim tag.

41.11.10 TPIU claim tag clear register (TPIU_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]:** Resets claim tag bits

Write:
0000: No effect
xxx1: Clears bit 0.
xx1x: Clears bit 1.
x1xx: Clears bit 2.
1xxx: Clears bit 3.
Read: Returns current value of claim tag.

41.11.11 TPIU device configuration register (TPIU_DEVIDR)

Address offset: 0xFC8

Reset value: 0x0000 0CA1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			Res.	MAXNUM[4:0]				
				r	r	r	r	r	r		r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SWOUARTNRZ:** Serial-wire output, NRZ supported
1: Supported

Bit 10 **SWOMAN:** Serial-wire output, Manchester encoded format supported
1: Supported

Bit 9 **TCLKDATA:** Indicates whether trace clock plus data is supported
0: Supported

Bits 8:6 **FIFOSIZE[2:0]:** FIFO size in powers of 2
0x2: FIFO size = 4 bytes

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **MAXNUM[4:0]:** Number/type of ATB input port multiplexing
0x0: One input port

41.11.12 TPIU device type identifier register (TPIU_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]:** Sub-classification
0x1: Trace port component

Bits 3:0 **MAJORTYPE[3:0]:** Major classification
0x1: Trace sink component

41.11.13 TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **F4KCOUNT[3:0]:** Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]:** JEP106 continuation code
0x4: Arm® JEDEC code

41.11.14 TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]:** Part number bits [7:0]
0x21: TPIU part number

41.11.15 TPIU CoreSight peripheral identity register 1 (TPIU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]:** JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]:** Part number bits [11:8]
0xD: TPIU part number

41.11.16 TPIU CoreSight peripheral identity register 2 (TPIU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]:** Component revision number
0x0: r0p0

Bit 3 **JEDEC:** JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]:** JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

41.11.17 TPIU CoreSight peripheral identity register 3 (TPIU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]:** Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]:** Customer modified
0x0: No customer modifications

41.11.18 TPIU CoreSight component identity register 0 (TPIU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]:** Component ID bits [7:0]
0x0D: Common ID value

41.11.19 TPIU CoreSight peripheral identity register 1 (TPIU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** Component ID bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]:** Component ID bits [11:8]
0x0: Common ID value

41.11.20 TPIU CoreSight component identity register 2 (TPIU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]:** Component ID bits [23:16]
0x05: Common ID value

41.11.21 TPIU CoreSight component identity register 3 (TPIU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]:** Component ID bits [31:24]
0xB1: Common ID value

41.11.22 TPIU register map and reset values

The TPIU registers are accessed by the debugger through AP1 at address range 0xE004 0000 to 0xE004 0FFC.

Table 418. TPIU register map and reset values

Offset	Register name	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
0x000	TPIU_SSPSR	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x004	TPIU_CSPSR	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x008 to 0x00C	Reserved																																
0x010	TPIU_ACPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x014 to 0x0EC	Reserved																																
0x0F0	TPIU_SPPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXMODE[1:0]	
	Reset value																															0	1
0x0F4 to 0x2FC	Reserved																																
0x300	TPIU_FFSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
	Reset value																													1	0	0	0
0x304	TPIU_FFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGIN	Res.	FONMAN	Res.	Res.	Res.	Res.	ENFCONT	Res.
	Reset value																								1		0					1	
0x308	TPIU_FSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSCOUNT[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x30C to 0xF9C	Reserved																																
0xFA0	TPIU_CLAIMSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													1	1	1	1
0xFA4	TPIU_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3:0]			
	Reset value																													0	0	0	0
0xFA8 to 0xFC4	Reserved																																
0xFC8	TPIU_DEVIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWONRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			Res.	MAXNUM[4:0]				
	Reset value																					1	1	0	0	1	0		0	0	0	0	1

Table 418. TPIU register map and reset values (continued)

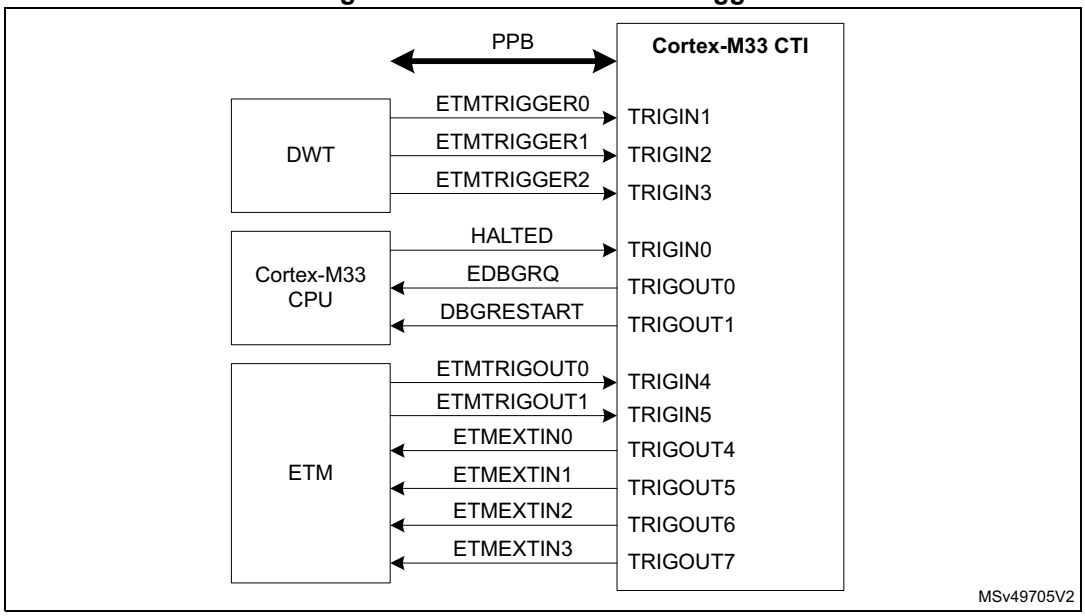
Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	TPIU_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE [3:0]				MAJORTYPE [3:0]			
	Reset value																									0	0	0	1	0	0	0	1
0xFD0	TPIU_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved																																
0xFE0	TPIU_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	TPIU_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	TPIU_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	TPIU_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	TPIU_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	TPIU_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	TPIU_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	TPIU_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 41.5: System debug AP0 features* for the register boundary addresses.

41.12 Cross trigger interface (CTI)

The CTI allows cross triggering between the processor and the ETM.

Figure 384. Embedded cross trigger



The CTI enables events from various sources to trigger the debug activity (for example halt the processor core).

The trigger input and output signals for the CTI are listed, respectively, in tables *419* and *420*.

Table 419. CTI inputs

No.	Source signal	Source component	Comments
0	HALTED	CPU	CPU halted - indicates CPU is in debug mode.
1	ETMTRIGGER0	DWT	DWT comparator output 0
2	ETMTRIGGER1	DWT	DWT comparator output 1
3	ETMTRIGGER2	DWT	DWT comparator output 2
4	ETMTRIGOUT0	ETM	ETM event output 0
5	ETMTRIGOUT1	ETM	ETM event output 1
6	-	-	Not used
7	-	-	Not used

Table 420. CTI outputs

No.	Source signal	Source component	Comments
0	EDBGRQ	CPU	CPU halt request - Puts CPU in debug mode.
1	DBGRESTART	CPU	CPU restart request - CPU exits debug mode.
2	-	-	Not used

Table 420. CTI outputs (continued)

No.	Source signal	Source component	Comments
3	-	-	Not used
4	ETMEXTIN0	ETM	ETM event input 0
5	ETMEXTIN1	ETM	ETM event input 1
6	ETMEXTIN2	ETM	ETM event input 2
7	ETMEXTIN3	ETM	ETM event input 3

For more information on the CTI and how to use it, refer to the Arm® CoreSight SoC-400 Technical Reference Manual *[2]*.

41.12.1 CTI registers

CTI control register (CTI_CONTROLR)

Address offset: 0x000

Reset value: 0x0000 0000

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

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **GLBEN:** Global cross-triggering enable
0: Disabled
1: Enabled

CTI trigger acknowledge register (CTI_INTACKR)

Address offset: 0x010

Reset value: 0x0000 0000

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **INTACK[7:0]:** Trigger acknowledge

There is one bit of the register for each CTI TRIGOUT output. When a 1 is written to a bit in this register, the corresponding CTI TRIGOUT output is acknowledged, causing the bit to be cleared.

CTI application trigger set register (CTI_APPSETR)

Address offset: 0x014

Reset value: 0x0000 0000

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPSET[3:0]:** Channel event setting

Read:
XXX0: Channel 0 event inactive
XXX1: Channel 0 event active
XX0X: Channel 1 event inactive
XX1X: Channel 1 event active
X0XX: Channel 2 event inactive
X1XX: Channel 2 event active
0XXX: Channel 3 event inactive
1XXX: Channel 3 event active
Write:
0000: No effect
XXX1: Sets event on channel 0.
XX1X: Sets event on channel 1.
X1XX: Sets event on channel 2.
1XXX: Sets event on channel 3.

CTI application trigger clear register (CTI_APPCLEAR)

Address offset: 0x018

Reset value: 0x0000 0000

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPCLEAR[3:0]:** Channel event clearing
0000: No effect
XXX1: Clears event on channel 0.
XX1X: Clears event on channel 1.
X1XX: Clears event on channel 2.
1XXX: Clears event on channel 3.

CTI application pulse register (CTI_APPPULSER)

Address offset: 0x01C

Reset value: 0x0000 0000

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPPULSE[3:0]:** Pulse channel event
This register clears itself immediately.
0000: No effect
XXX1: Generates pulse on channel 0.
XX1X: Generates pulse on channel 1.
X1XX: Generates pulse on channel 2.
1XXX: Generates pulse on channel 3.

CTI trigger in x enable register (CTI_INENRx)

Address offset: 0x020 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGINEN[3:0]:** cross trigger event enable/disable

Enables or disables a cross trigger event on each of the four channels when CTITRIGINx is activated (x = 0 to 7).

0000: Trigger does not generate events on any channel.

XXX1: Trigger x generates events on channel 0.

XX1X: Trigger x generates events on channel 1.

X1XX: Trigger x generates events on channel 2.

1XXX: Trigger x generates events on channel 3.

CTI trigger out x enable register (CTI_OUTENRx)

Address offset: 0x0A0 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGOUTEN[3:0]**: For each channel, defines whether an event on that channel generates a trigger on CTI TRIGOUTx (x = 0 to 7).

0000: Channel events do not generate triggers on any trigger output.

XXX1: Channel 0 events generate triggers on trigger output x.

XX1X: Channel 1 events generate triggers on trigger output x.

X1XX: Channel 2 events generate triggers on trigger output x.

1XXX: Channel 3 events generate triggers on trigger output x.

CTI trigger in status register (CTI_TRGISTSR)

Address offset: 0x130

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINSTATUS[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGINSTATUS[7:0]:** Trigger input status

There is one bit of the register for each CTITRIGIN input. When a bit is set to 1, it indicates that the corresponding trigger input is active. When it is set to 0, the corresponding trigger input is inactive.

CTI trigger out status register (CTI_TRGOSTSR)

Address offset: 0x134

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTSTATUS[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGOUTSTATUS[7:0]:** Trigger output status

There is one bit of the register for each CTI TRIGOUT output. When a bit is set to 1, it indicates that the corresponding trigger output is active. When it is set to 0, the corresponding trigger output is inactive.

CTI channel in status register (CTI_CHINSTSR)

Address offset: 0x138

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHINSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHINSTATUS[3:0]:** Channel input status

There is one bit of the register for each channel input. When a bit is set to 1, it indicates that the corresponding channel input is active. When it is set to 0, the corresponding channel input is inactive.

CTI channel out status register (CTI_CHOUTSTSR)

Address offset: 0x13C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHOUTSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHOUTSTATUS[3:0]:** Channel output status

There is one bit of the register for each channel output. When a bit is set to 1, it indicates that the corresponding channel output is active. When it is set to 0, the corresponding channel output is inactive.

CTI channel gate register (CTI_GATER)

Address offset: 0x140

Reset value: 0x0000 000F

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **GATEEN[3:0]:** Channel output enable

For each channel, defines whether an event on that channel can propagate over the CTM to other CTIs.
0000: Channels events do not propagate.
XXX1: Channel 0 events propagate.
XX1X: Channel 1 events propagate.
X1XX: Channel 2 events propagate.
1XXX: Channel 3 events propagate.

CTI device configuration register (CTI_DEVIDR)

Address offset: 0xFC8

Reset value: 0x0004 0800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH[3:0]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMTRIG[7:0]								Res.	Res.	Res.	EXTMUXNUM[4:0]				
r	r	r	r	r	r	r	r				r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **NUMCH[3:0]:** Number of ECT channels

0x4: 4 channels

Bits 15:8 **NUMTRIG[7:0]:** Number of ECT triggers
0x8: 8 trigger inputs and 8 trigger outputs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **EXTMUXNUM[4:0]:** Number of trigger input/output multiplexers
0x0: None

CTI device type identifier register (CTI_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]:** Sub-classification
0x1: Indicates that this component is a cross-triggering one.

Bits 3:0 **MAJORTYPE[3:0]:** Major classification
0x4: Indicates that this component allows a debugger to control other components in a CoreSight™ SoC-400 system.

CTI CoreSight peripheral identity register 4 (CTI_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **F4KCOUNT[3:0]:** Register file size
0x0: The register file occupies a single 4-Kbyte region

Bits 3:0 **JEP106CON[3:0]:** JEP106 continuation code
0x4: Arm® JEDEC code

CTI CoreSight peripheral identity register 0 (CTI_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]:** Part number bits [7:0]
0x21: CTI part number

CTI CoreSight peripheral identity register 1 (CTI_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]:** JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]:** Part number bits [11:8]
0xD: CTI part number

CTI CoreSight peripheral identity register 2 (CTI_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]:** Component revision number
0x0: r0p0

Bit 3 **JEDEC:** JEDEC assigned value
0x1: Designer identifier specified by JEDEC

Bits 2:0 **JEP106ID[6:4]:** JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

CTI CoreSight peripheral identity register 3 (CTI_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]:** Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]:** Customer modified
0x0: No customer modifications

CTI CoreSight component identity register 0 (CTI_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]:** Component ID bits [7:0]
0x0D: Common ID value

CTI CoreSight peripheral identity register 1 (CTI_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** Component ID bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]:** Component ID bits [11:8]
0x0: Common ID value

CTI CoreSight component identity register 2 (CTI_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]:** Component ID bits [23:16]
0x05: Common ID value

CTI CoreSight component identity register 3 (CTI_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]:** Component ID bits [31:24]
0xB1: Common ID value

CTI register map and reset values

The CTI registers are accessed by the debugger through AP1 at address range 0xE004 2000 to 0xE004 2FFC.

Table 421. CTI register map and reset values

Offset	Register name	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
0x000	CTI_CONTROLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLBEN
	Reset value																																0
0x004 to 0x00C	Reserved																																
0x010	CTI_INTACKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x014	CTI_APPSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPSET[3:0]			
	Reset value																													0	0	0	0
0x018	CTI_APPCLEAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPCLEAR [3:0]			
	Reset value																													0	0	0	0
0x01C	CTI_APPPULSER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPPULSE [3:0]			
	Reset value																													0	0	0	0
0x020 to 0x03C	CTI_INENR0 to CTI_INENR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN [3:0]			
	Reset value																													0	0	0	0
0x040 to 0x09C	Reserved																																
0x0A0 to 0x0BC	CTI_OUTENR0 to CTI_OUTENR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN [3:0]			
	Reset value																													0	0	0	0
0x0C0 to 0x12C	Reserved																																
0x130	CTI_TRIGISTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINSTATUS[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x134	CTI_TRIGOSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTSTATUS[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x138	CTI_CHINSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHIN STATUS[3:0]			
	Reset value																													0	0	0	0
0x13C	CTI_CHOUTSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHOUT STATUS[3:0]			
	Reset value																													0	0	0	0
0x140	CTI_GATER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GATEEN[3:0]			
	Reset value																													1	1	1	1
0x144 to 0xFC4	Reserved																																
0xFC8	CTI_DEVIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH[3:0]				NUMTRIG[7:0]								Res.	Res.	Res.	EXTMUXNUM [4:0]				
	Reset value													0	1	0	0	0	0	0	0	1	0	0	0				0	0	0	0	0

Table 421. CTI register map and reset values (continued)

Offset	Register name	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
0xFCC	CTI_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE [3:0]				MAJORTYPE [3:0]			
	Reset value																									0	0	0	1	0	1	0	0
0xFD0	CTI_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved																																
0xFE0	CTI_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	CTI_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	CTI_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	CTI_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	CTI_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	CTI_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	CTI_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	CTI_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 41.5: System debug AP0 features* for the register boundary addresses.

41.13 Microcontroller debug unit (DBGMCU)

DBGMCU is a component containing a number of registers that control the power and clock behavior in debug mode. It allows the debugger (or the debug software) to perform the following tasks:

- Maintain the clock and power to the processor core and system debug and trace components when in low-power modes (Stop or Standby).
- Stop the clock to certain peripherals (such as watchdogs, timers, RTC) when the processor core is halted in debug mode.
- Obtain information on the processor core and system operating modes.

41.13.1 DBGMCU access

The DBGMCU can be accessed through both AP0 and AP1, and directly by the Cortex-M33 processor with the following restrictions:

- When there is no debugger connected to AP0:
  - DBGMCU can be accessed through AP1 and by the Cortex-M33 processor.
- When the debugger is connected to AP0:
  - DBGMCU can only be accessed through AP0.
  - DBGMCU accesses by the Cortex-M33 processor generate a bus fault.

41.13.2 Device ID

The DBGMCU includes an identity code register, DBGMCU_IDCODE. This register contains the ID code for the device. The debug tools can locate this register via the CoreSight discovery procedure described in *Section 41.6.1: CPU ROM tables*.

41.13.3 Part number codification

The part number codification can be read from the DBGMCU registers DBGMCU_PNCR.

41.13.4 Low-power mode emulation

When the device enters Sleep mode, debugger access to SRAMs can be kept by setting DBG_SLEEP.

When the device enters Stop mode or Standby mode, the debugger can no longer access the CPU debug access port and loses the connection with the device CPU. In Stop mode it remains still possible to access the DBGMCU from the debugger. To gain or keep CPU debug access, the debugger (or software) can set the DBG_STOP and/or DBG_STANDBY bits in the DBGMCU_CR register. This allows maintaining the clock and the power to the device CPU and DBGMCU even when it enters Stop mode or Standby mode. When in debug Stop or debug Standby mode, the processor remains in Deepsleep mode and exit this mode in the normal way. In these debug modes the processor can also be woken up by a debugger halt instruction, which puts the device back to Run mode. When in debug Stop or debug Standby mode all other device peripherals behave as in Stop mode or Standby mode and are not accessible from the debugger. In debug Stop mode, the autonomous peripherals continue to operate when they request their clock. In debug Standby mode the peripherals are kept under reset.

Note: When using debug Stop mode (DBG_STOP) a SysTick event wakes up the device. It is good practice that the software disables the SysTick, before the CPU enters Deepsleep.

In Stop mode, without enabling DBG_STOP, debug access via the DAP is still possible to the DBGMCU. To keep access to the DBGMCU in Standby mode, the DBG_STANDBY needs to be enabled. Keeping access to the DBGMCU allows access to the DBGMCU registers.

In debug Stop mode and debug Standby mode the device behavior does not differ from Stop mode and Standby mode, with the exception of SRAMs, which are retained and keep their values. For an overview see *Table 422*.

Table 422. Low power debug overview

<table>
<tr><th>Low power mode</th><th>System clock</th><th>Power supply</th><th>DBGMCU and DAP</th><th>CPU debug</th><th>Autonomous peripherals</th><th>Other peripherals</th><th>SRAMs</th><th>Low-power mode exit</th><th>Debug wake-up</th></tr>
<tr><td>DBG_SLEEP</td><td>Any</td><td rowspan="4">On</td><td colspan="2" rowspan="2">Accessible</td><td colspan="2">As Sleep mode</td><td>Accesible</td><td>Interrupt or event</td><td>Halt CPU</td></tr>
<tr><td>DBG_STOP(1)</td><td rowspan="2">HSI16</td><td>As Stop mode, clock on request</td><td>As Stop mode, clock off</td><td>As Stop mode retained</td><td>Stop wake-up</td><td>Halt CPU</td></tr>
<tr><td>DBG_STANDBY(1)</td><td colspan="2">As Standby mode Under reset</td><td>As Standby mode retained</td><td>Standby reset</td><td>Halt CPU</td></tr>
<tr><td>Stop(2)</td><td rowspan="3">Off</td><td rowspan="2">Accessible</td><td>As Stop mode, clock off</td><td>As Stop mode, clock on request</td><td>As Stop mode, clock off</td><td>As Stop mode retained</td><td>Stop wake-up</td><td>Via DBG_STOP</td></tr>
<tr><td>Standby with CDBGPWRUPREQ(2)</td><td colspan="3">Under reset</td><td>As Standby mode retained</td><td>Standby reset</td><td>Via DBG_ STANDBY</td></tr>
<tr><td>Standby(3)</td><td>Off</td><td colspan="4">Power down</td><td>Power down when not retained</td><td>Standby reset</td><td>None</td></tr>
</table>

1. In DBG_STOP and DBG_STANDBY mode the debug connection with the CPU and access to the DBGMCU is kept. CPU can be woken up with debugger halt.
2. In Stop mode and Standby mode with CDBGPWRUPREQ, only debug access to the DBGMCU is kept, the debug connection to the CPU is lost. DBGMCU can be used to activate the CPU clock via DBG_STOP or DBG_STANDBY.
3. In Standby mode all debug accesses are lost.

In Stop debug mode and Standby debug mode both the CPU debug and DBGMCU are accessible from the debugger. A debugger halt to the CPU wakes up the CPU and sets the system to Run mode. Only in Run mode the peripherals with their bus clock enabled are accessible from the debugger.

In Stop mode with CDBGPWRUPREQ enabled, only the DBGMCU is accessible from the debugger. Via the DBGMCU DBG_STOP the debugger can move the device to the debug Stop mode to regain debug access to the CPU.

In Standby mode all debug accesses are lost.

41.13.5 Low-power mode status

The DBGMCU provides status flags for the processor core and device operating mode in DBGMCU_SCR.

Table 423. Low-power mode status flags

Status flag	Description
CS	CPU in Sleep mode
CDS	CPU in Deepsleep mode
STOPF	Device in Stop mode
SBF	Device in Standby mode (this bit is only available with DBG_STANDBY or CDBGPWRUPREQ)

41.13.6 Peripheral clock freeze

The DBGMCU provides a peripheral clock freeze feature in the DBGMCU_zzzzFZR registers. This allows the operation (clock) of certain peripherals to be suspended in debug mode (when the processor is halted). Peripherals supporting this feature are listed in *Table 424*.

Table 424. Peripheral clock freeze control

Bus	Control	Peripheral	Behavior
APB1L	DBGMCU_APB1LFZR	I2C1	Freeze timeout counting
		IWDG	Freeze counting
		TIM2	
APB1H	DBGMCU_APB1HFZR	LPTIM2	Freeze counting
APB2	DBGMCU_APB2HFZR	TIM17	Freeze counting
		TIM16	
APB7	DBGMCU_APB7FZR	RTC	Freeze counting
		LPTIM1	
		I2C3	Freeze timeout counting
AHB1	DBGMCU_AHB1FZR	LPDMA1	Freeze channel transfers

Each peripheral unit or DMA channel has a corresponding control bit, DBG_xxx_STOP, where xxx is the acronym of the peripheral (or DMA channel). The control bits are organized in DBGMCU_zzzzFZR registers, where zzzz corresponds to the name of the bus (AHB or APB). For example, DBGMCU_APB1LFZR contains the control bits for peripherals on the APB1L bus.

The control bit, when set, causes the corresponding peripheral operation to be suspended when the CPU is stopped in debug (HALTED = 1), according to the table below:

Table 425. Peripheral behavior in debug mode

HALTED	DBG_xxx_STOP	Peripheral behaviour
0	X	The operation continues.
1	0	The operation continues.
1	1	The operation is suspended.

The accessibility of DBG_xxx_STOP bits by the debugger depends on the state of the authentication signal spiden.

When spiden = 1 (secure privilege debug enabled), all bits can be modified by a secure access. Only bits corresponding to nonsecure peripherals (or DMA channels) can be modified by a nonsecure access. All bits can be read by both nonsecure or secure accesses.

When spiden = 0 (secure privilege debug disabled), only nonsecure accesses are possible (secure access requests by the debugger are converted to nonsecure by the CPU). Only bits corresponding to nonsecure peripherals (or DMA channels) can be modified. All bits can be read. This is summarized in the table below.

Table 426. Debugger access to freeze register bits

spiden	Peripheral xxx status	Access security attribute	DBG_xxx_STOP can be modified?	DBG_xxx_STOP can be read?
0	Nonsecure	Nonsecure	Yes	Yes
		Secure	Yes(1)	Yes(1)
	Secure	Nonsecure	No	Yes
		Secure	No(1)	Yes(1)
1	Nonsecure	Nonsecure	Yes	Yes
		Secure	Yes	Yes
	Secure	Nonsecure	No	Yes
		Secure	Yes	Yes

1. When spiden = 0, secure access requests by the debugger are converted to nonsecure.

The status (secure or nonsecure) of a TrustZone–aware peripheral or a DMA channel, is signaled to the DBGMCU by the peripheral.

The CPU access to the DBG_xxx_STOP bits does not depend on spiden. This access depends only on the security status of the peripheral. The bits corresponding to a secure peripheral (or DMA channel) can only be modified by a secure access (when the CPU is in secure state).

41.13.7 DBGMCU registers

The DBGMCU registers are not reset by a system reset, only by a power-on reset.

DBGMCU identity code register (DBGMCU_IDCODE)

Address offset: 0x000

Reset value: 0xXXXX 64B2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DEV_ID[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV_ID[15:0]:** Revision ID
0x1000: Revision A

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]:** Device ID
0x4B2: STM32WBA2xxx

DBGMCU status and configuration register (DBGMCU_SCR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CDS	CS	Res.	Res.	Res.	SBF	STOPF	LPMS[2:0]		
						r	r				r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACE_MODE[1:0]		TRACE_EN	TRACE_IOEN	Res.	DBG_STANDBY	DBG_STOP	DBG_SLEEP
								rw	rw	rw	rw		rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **CDS:** CPU Deepsleep
0: CPU not in Deepsleep
1: CPU in Deepsleep

Bit 24 **CS:** CPU Sleep
0: CPU not in Sleep
1: CPU in Sleep

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **SBF:** Device Standby flag
0: Device not in Standby mode
1: Device in Standby mode

Bit 19 **STOPF:** Device Stop flag
0: Device not in Stop mode
1: Device in Stop mode

Bits 18:16 **LPMS[2:0]:** Device low power mode selected
000: Stop 0 mode
001: Stop 1 mode
010: Stop 2 mode
011: Stop 3 mode
10x: Standby mode
Others reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **TRACE_MODE[1:0]:** Trace pin assigment
00: Trace pins assigned for asynchronous mode (TRACESWO)
01: Trace pins assigned for synchronous mode with port width of 1 (TRACECK, TRACED0)
10: Trace pins assigned for synchronous mode with port width of 2 (TRACECK, TRACED1:0)
11: Trace pins assigned for synchronous mode with port width of 4 (TRACECK, TRACED3:0)

Bit 5 **TRACE_EN:** trace port enable
This bit enables the trace port clock (TRACECK).
0: Disabled
1: Enabled

Bit 4 **TRACE_IOEN:** trace port pins enable
This bit enables the trace port clock (TRACECK).
0: Disabled - trace port pins not assigned
1: Enabled - trace port pins assigned according to the selection in TRACE_MODE

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DBG_STANDBY:** Allows debug in Standby mode
0: Normal Standby mode operation, all clocks are disabled automatically in Stop mode and the core is powered down.
1: Core power maintained and CPU debug clock enabled in Standby mode, all other are under reset in Stop mode, except for DBGMCU.
The CPU debug and DBGMCU clocks remain active and the HSI16 oscillator is used as system clock. The supply and SRAM memory content is maintained during Standby debug mode, allowing CPU debug capability. On exit from Standby mode, a standby reset is performed.

Bit 1 **DBG_STOP:** Allows debug in Stop mode
0: Normal Stop mode operation, all clocks are disabled automatically in Stop mode.
1: CPU debug clock enabled in Stop mode, all other peripheral clocks are disabled automatically in Stop mode, except for DBGMCU.
The CPU debug and DBGMCU clocks remain active and the HSI16 oscillator is used as system clock during Stop debug mode, allowing CPU debug capability. On exit from Stop mode, the clock settings are set to the Stop mode exit state.

Bit 0 **DBG_SLEEP:** Allows SRAM access in Sleep mode
0: Normal Sleep mode operation, all clocks are disabled automatically in Sleep mode.
1: SRAM clocks enabled in Sleep mode, all other peripheral clocks are disabled automatically in Sleep mode.
When DBG_STOP is also set, allow SRAM access in Stop mode.
When DBG_STANDBY is also set, allow SRAM access in Standby mode.

DBGMCU APB1L peripheral freeze register (DBGMCU_APB1LFZR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.
										rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBG_IWDG_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM2_STOP
			rw												rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **DBG_I2C1_STOP:** I2C1 SMBUS timeout stop in CPU debug

0: Normal operation. I2C1 SMBUS timeout continues to operate while CPU is in debug mode.

1: Stop in debug. I2C1 SMBUS timeout is frozen while CPU is in debug mode.

Bits 20:13 Reserved, must be kept at reset value.

Bit 12 **DBG_IWDG_STOP:** IWDG stop in CPU debug

0: Normal operation. IWDG continues to operate while CPU is in debug mode.

1: Stop in debug. IWDG is frozen while CPU is in debug mode.

Bits 11:1 Reserved, must be kept at reset value.

Bit 0 **DBG_TIM2_STOP:** TIM2 stop in CPU debug

0: Normal operation. TIM2 continues to operate while CPU is in debug mode.

1: Stop in debug. TIM2 is frozen while CPU is in debug mode.

DBGMCU APB1H peripheral freeze register (DBGMCU_APB1HFZR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM2_STOP	Res.	Res.	Res.	Res.	Res.
										rw					

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **DBG_LPTIM2_STOP**: LPTIM2 stop in CPU debug

0: Normal operation. LPTIM2 continues to operate while CPU is in debug mode.

1: Stop in debug. LPTIM2 is frozen while CPU is in debug mode.

Bits 4:0 Reserved, must be kept at reset value.

DBGMCU APB2 peripheral freeze register (DBGMCU_APB2FZR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	Res.
													rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **DBG_TIM17_STOP**: TIM17 stop in CPU debug
0: Normal operation. TIM17 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM17 is frozen while CPU is in debug mode.

Bit 17 **DBG_TIM16_STOP**: TIM16 stop in CPU debug
0: Normal operation. TIM16 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM16 is frozen while CPU is in debug mode.

Bits 16:0 Reserved, must be kept at reset value.

DBGMCU APB7 peripheral freeze register (DBGMCU_APB7FZR)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM1_STOP	Res.
	rw													rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DBG_I2C3_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					rw										

Bit 31 Reserved, must be kept at reset value.

Bit 30 **DBG_RTC_STOP**: RTC stop in CPU debug
0: Normal operation. RTC continues to operate while CPU is in debug mode.
1: Stop in debug. RTC is frozen while CPU is in debug mode.

Bits 29:18 Reserved, must be kept at reset value.

Bit 17 **DBG_LPTIM1_STOP**: LPTIM1 stop in CPU debug
0: Normal operation. LPTIM1 continues to operate while CPU is in debug mode.
1: Stop in debug. LPTIM1 is frozen while CPU is in debug mode.

Bits 16:11 Reserved, must be kept at reset value.

Bit 10 **DBG_I2C3_STOP**: I2C3 SMBUS timeout stop in CPU debug
0: Normal operation. I2C3 SMBUS timeout continues to operate while CPU is in debug mode.
1: Stop in debug. I2C3 SMBUS timeout is frozen while CPU is in debug mode

Bits 9:0 Reserved, must be kept at reset value.

DBGMCU AHB1 peripheral freeze register (DBGMCU_AHB1FZR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPDMA1_CH7_STOP	DBG_LPDMA1_CH6_STOP	DBG_LPDMA1_CH5_STOP	DBG_LPDMA1_CH4_STOP	DBG_LPDMA1_CH3_STOP	DBG_LPDMA1_CH2_STOP	DBG_LPDMA1_CH1_STOP	DBG_LPDMA1_CH0_STOP
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **DBG_LPDMA1_CH7_STOP**: LPDMA 1 channel 7 stop in CPU debug
0: Normal operation. LPDMA 1 channel 7 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA 1 channel 7 is frozen while CPU is in debug mode.

Bit 6 **DBG_LPDMA1_CH6_STOP**: LPDMA 1 channel 6 stop in CPU debug
0: Normal operation. LPDMA 1 channel 6 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA 1 channel 6 is frozen while CPU is in debug mode.

Bit 5 **DBG_LPDMA1_CH5_STOP**: LPDMA 1 channel 5 stop in CPU debug
0: Normal operation. LPDMA 1 channel 5 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA 1 channel 5 is frozen while CPU is in debug mode.

Bit 4 **DBG_LPDMA1_CH4_STOP**: LPDMA 1 channel 4 stop in CPU debug
0: Normal operation. LPDMA 1 channel 4 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA 1 channel 4 is frozen while CPU is in debug mode.

Bit 3 **DBG_LPDMA1_CH3_STOP**: LPDMA 1 channel 3 stop in CPU debug
0: Normal operation. LPDMA 1 channel 3 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA 1 channel 3 is frozen while CPU is in debug mode.

Bit 2 **DBG_LPDMA1_CH2_STOP**: LPDMA 1 channel 2 stop in CPU debug
0: Normal operation. LPDMA 1 channel 2 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA 1 channel 2 is frozen while CPU is in debug mode.

Bit 1 **DBG_LPDMA1_CH1_STOP**: LPDMA 1 channel 1 stop in CPU debug
0: Normal operation. LPDMA 1 channel 1 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA 1 channel 1 is frozen while CPU is in debug mode.

Bit 0 **DBG_LPDMA1_CH0_STOP**: LPDMA 1 channel 0 stop in CPU debug
0: Normal operation. LPDMA 1 channel 0 continues to operate while CPU is in debug mode.
1: Stop in debug. LPDMA 1 channel 0 is frozen while CPU is in debug mode.

DBGMCU status register (DBGMCU_SR)

Address offset: 0x0FC

Reset value: 0x000X 0003

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

Bits 31:16 **AP_ENABLED[15:0]**: Bit n identifies whether access port APn is open

This bit can be accessed via the debug port or locked (debug access to the APn is blocked, except for DBGMCU access)

Bit n = 0: APn blocked (except for access to DBGMCU)

Bit n = 1: APn enabled

Bits 15:0 **AP_PRESENT[15:0]**: Bit n identifies whether access port APn is present in device

Bit n = 0: APn absent

Bit n = 1: APn present

DBGMCU debug host authentication register (DBGMCU_DBG_AUTH_HOST)

Address offset: 0x100

Reset value: 0xXXXX XXXX

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

Bits 31:0 **AUTH_KEY[31:0]**: Device authentication key

The device specific 128-bit authentication key (OEMn key) must be written to this register (in four successive 32-bit writes, least significant word first) to permit RDP regression. Writing a wrong key locks access to the device and prevents code execution from the flash memory.

DBGMCU debug device authentication register (DBGMCU_DBG_AUTH_DEVICE)

Address offset: 0x104

Reset value: 0xXXXX XXXX

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

Bits 31:0 **AUTH_ID[31:0]**: Device authentication key

The device specific 128-bit authentication key (OEMn key) must be written to this register (in four successive 32-bit writes, least significant word first) to permit RDP regression. Writing a wrong key locks access to the device and prevents code execution from the flash memory.

DBGMCU part number codification register (DBGMCU_PNCR)

Address offset: 0x7DC

Reset value: 0xXXXX XXXX

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

Bits 31:0 **CODIFICATION[31:0]:** Part number codification

0x0033 3241: STMicroelectronics STM32WBA23xx part number codification.

0x0035 3241: STMicroelectronics STM32WBA25xx part number codification.

DBGMCU CoreSight peripheral identity register 4 (DBGMCU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **F4KCOUNT[3:0]:** Register file size
0x0: The register file occupies a single 4-Kbyte region

Bits 3:0 **JEP106CON[3:0]:** JEP106 continuation code
0x0: STMicroelectronics JEDEC code

DBGMCU CoreSight peripheral identity register 0 (DBGMCU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]:** Part number bits [7:0]
0x00: DBGMCU part number

DBGMCU CoreSight peripheral identity register 1 (DBGMCU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]:** JEP106 identity code bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]:** Part number bits [11:8]
0x0: DBGMCU part number

DBGMCU CoreSight peripheral identity register 2 (DBGMCU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]:** Component revision number
0x0: r0p0

Bit 3 **JEDEC:** JEDEC assigned value
0x1: Designer identifier specified by JEDEC

Bits 2:0 **JEP106ID[6:4]:** JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

DBGMCU CoreSight peripheral identity register 3 (DBGMCU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]:** Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]:** Customer modified
0x0: No customer modifications

DBGMCU CoreSight component identity register 0 (DBGMCU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]:** Component ID bits [7:0]
0x0D: Common ID value

DBGMCU CoreSight peripheral identity register 1 (DBGMCU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00F0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** Component ID bits [15:12] - component class
0xF: No CoreSight component

Bits 3:0 **PREAMBLE[11:8]:** Component ID bits [11:8]
0x0: Common ID value

DBGMCU CoreSight component identity register 2 (DBGMCU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]:** Component ID bits [23:16]
0x05: Common ID value

DBGMCU CoreSight component identity register 3 (DBGMCU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]:** Component ID bits [31:24]
0xB1: Common ID value

41.13.8 DBGMCU register map and reset values

Table 427. DBGMCU register map and reset values

Offset	Register name	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
0x000	DBGMCU_IDCODE	REV_ID[15:0]																Res.	Res.	Res.	Res.	DEV_ID[11:0]											
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x004	DBGMCU_SCR	Res.	Res.	Res.	Res.	Res.	Res.	CDS	CS	Res.	Res.	Res.	SBF	STOPF	LPMS[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACE_MODE[1:0]		TRACE_EN	TRACE_IOEN	Res.	DBG_STANDBY	DBG_STOP	DBG_SLEEP
	Reset value							0	0				0	0	0	0	0									0	0	0	0		0	0	0
0x008	DBGMCU_APB1LFZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_IWDG_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM2_STOP
	Reset value											0									0												0
0x00C	DBGMCU_APB1HFZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM2_STOP	Res.	Res.	Res.	Res.	Res.
	Reset value																											0					
0x010	DBGMCU_APB2FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0																	
0x014 to 0x020	Reserved																																
0x024	DBGMCU_APB7FZR	Res.	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C3_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0													0							0										
0x028	DBGMCU_AHB1FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPDMA1_CH7_STOP	DBG_LPDMA1_CH6_STOP	DBG_LPDMA1_CH5_STOP	DBG_LPDMA1_CH4_STOP	DBG_LPDMA1_CH3_STOP	DBG_LPDMA1_CH2_STOP	DBG_LPDMA1_CH1_STOP	DBG_LPDMA1_CH0_STOP
	Reset value																									0	0	0	0	0	0	0	0

Table 427. DBGMCU register map and reset values (continued)

Offset	Register name	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
0x02C to 0x0F8	Reserved																																
0x0FC	DBGMCU_SR	AP_ENABLED[15:0]																AP_PRESENT[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0x100	DBGMCU_DBG_AUTH_HOST	AUTH_KEY[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x104	DBGMCU_DBG_AUTH_DEVICE	A UTH_KEY[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x108 to 0x7D8	Reserved																																
0x7DC	DBGMCU_PNCR	CODIFICATION[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x7E0 to 0xFCC	Reserved																																
0xFD0	DBGMCU_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4 to 0xFDC	Reserved																																
0xFE0	DBGMCU_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0xFE4	DBGMCU_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									0	0	0	0	0	0	0	0
0xFE8	DBGMCU_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	0
0xFEC	DBGMCU_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	DBGMCU_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	DBGMCU_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	1	1	1	0	0	0	0
0xFF8	DBGMCU_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	DBGMCU_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 41.5.1: System debug ROM table* and *Section 41.6.1: CPU ROM tables* for the register boundary addresses.

41.14 References

[1] IHI 0031C (ID080813) - Arm® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2, Issue C, 8th Aug 2013

[2] DDI 0480F (ID100313) - Arm® CoreSight™ SoC-400 r3p2 Technical Reference Manual, Issue G, 16th March 2015

[3] DDI 0314H - Arm® CoreSight™ Components Technical Reference Manual, Issue H, 10 July, 2009

[4] DDI 0553A (ID092917) - Arm®v8-M Architecture Reference Manual, Issue A.f, 29 September 2017

[5] 100230_0002_00_en - Arm® Cortex®-M33 Processor r0p2 Technical Reference Manual, Issue 0002-00, 10 May 2017

[6] 100232_0001_00_en - Arm® Sight ETM-M33 r0p1 Technical Reference Manual, Issue 0001-00, 3 February 2017

42 Device electronic signature (DESIG)

The device electronic signature is stored in the system memory area of the flash memory module, and can be read using the debug interface or by the CPU. It contains factory-programmed identification and calibration data that allow the user firmware or other external devices to match the characteristics of the microcontroller automatically.

42.1 Device electronic signature registers

42.1.1 DESIG package data register (DESIG_PKGR)

Address offset: 0x000

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKG[4:0]				
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PKG[4:0]**: Package type
00001: UFQFPN32
00010: UFQFPN48
00011: WLCSP37
00101:UFQFPN32 USB
00110: UFQFPN48 USB
01101: WLCSP37 SMPS USB
01110: UFQFPN48 SMPS USB
Others: reserved

42.1.2 DESIG 96-bit unique device ID register 1 (DESIG_UIDR1)

Address offset: 0x200

Reset value: 0xXXXX XXXX (X is factory-programmed)

The 96-bit unique device identifier is ideally suited:

- For use as a serial number (USB string serial number or other end applications).
- For use as part of the security keys, to increase the code security in the flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the memory.
- During processes such as secure boot.

This unique device identifier provides a reference number, unique for a given device and in any context. These bits cannot be altered by the user.

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

Bits 31:0 **UID[31:0]:** X and Y coordinates on the wafer expressed in BCD format

42.1.3 DESIG 96-bit unique device ID register 2 (DESIG_UIDR2)

Address offset: 0x204

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[63:32]:** Lot and wafer number
UID[63:40] = LOT_NUM[23:0], lot number lower part (ASCII encoded)
UID[39:32] = WAF_NUM[7:0], wafer number (8-bit unsigned number)

42.1.4 DESIG 96-bit unique device ID register 3 (DESIG_UIDR3)

Address offset: 0x208

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[95:64]:** LOT_NUM[55:24], lot number higher part (ASCII encoded)

42.1.5 DESIG temperature calibration 1 register (DESIG_TSCAL1R)

Address offset: 0x210

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TS_CAL1[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **TS_CAL1[11:0]:** Factory temperature sensor calibration 1 value for ADC4

42.1.6 DESIG temperature calibration 2 register (DESIG_TSCAL2R)

Address offset: 0x240

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TS_CAL2[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **TS_CAL2[11:0]:** Factory temperature sensor calibration 2 value for ADC4

Bits 15:0 Reserved, must be kept at reset value.

42.1.7 DESIG flash size data register (DESIG_FLASHSIZER)

Address offset: 0x2A0

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RAM_SIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_SIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **RAM_SIZE[15:0]:** RAM memory size

These bits indicate the size of the total device system RAM memory in Kbytes.
For example, 0x0040 corresponds to 64 Kbytes.

Bits 15:0 **FLASH_SIZE[15:0]:** Flash memory size
These bits indicate the size of the device Flash memory in Kbytes.
As an example, 0x0040 corresponds to 64 Kbytes.

42.1.8 DESIG internal voltage reference calibration register (DESIG_VREFINTCALR)

Address offset: 0x2A4

Reset value: 0xXXXX XXXX (X is factory-programmed)

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:8 **VREFINT_CAL[11:0]:** Factory internal voltage reference calibration value for ADC4

Bits 7:0 Reserved, must be kept at reset value.

42.1.9 DESIG IEEE 64-bit unique device ID register 1 (DESIG_UID64R1)

Address offset: 0x500

Reset value: 0xXXXX XXXX (X is factory-programmed)

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

Bits 31:0 **DEVNUM[31:0]:** device number
The 32-bit unique device number is a sequential number, different for each individual device.

42.1.10 DESIG IEEE 64-bit unique device ID register 2 (DESIG_UID64R2)

Address offset: 0x504

Reset value: 0x0080 E120

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

Bits 31:8 **STID[23:0]:** company ID
0x0080E1 for STMicroelectronics

Bits 7:0 **DEVID[7:0]:** device ID
0x20: STM32WBA2xxx

42.1.11 DESIG register map

Table 428. DESIG register map and reset values

Offset	Register	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
0x000	**DESIG_ PKGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKG[4:0]				
	Reset value																												x	x	x	x	x
0x004-0x1FF	**Reserved**																																
0x200	**DESIG_ UIDR1**	UID[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x204	**DESIG_ UIDR2**	UID[63:32]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x208	**DESIG_ UIDR3**	UID[95:64]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x210	**DESIG_ TSCAL1R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS_CAL1[11:0]											
	Reset value																					x	x	x	x	x	x	x	x	x	x	x	x
0x214-0x23F	**Reserved**																																
0x240	**DESIG_ TSCAL2R**	Res.	Res.	Res.	Res.	TS_CAL2[11:0]												Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					x	x	x	x	x	x	x	x	x	x	x	x																
0x248-0x29F	**Reserved**																																
0x2A0	**DESIG_ FLASHSIZER**	RAM_SIZE[15:0]																FLASH_SIZE[15:0]															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x2A4	**DESIG_ VREFINT CALR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VREFINT_CAL[11:0]												Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													x	x	x	x	x	x	x	x	x	x	x	x								
0x2A8-0x4FF	**Reserved**																																
0x500	**DESIG_ UID64R1**	DEVNUM[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x504	**DESIG_ UID64R2**	STID[23:0]																								DEVID[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

43 Revision history

Table 429. Document revision history

Date	Revision	Changes
23-Feb-2026	1	Initial release.

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

The purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of the purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers’ market segment, the purchasers shall contact ST for more information.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to *www.st.com/trademarks*. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2026 STMicroelectronics – All rights reserved